Hierarchical Node Clustering in Polymorphic Processor Arrays

Massimo Maresca and Hungwen Li

TR-91-042

July, 1991
HIERARCHICAL NODE CLUSTERING  
IN POLYMORPHIC PROCESSOR ARRAYS

Massimo Maresca  
International Computer Science Institute  
Berkeley, California

Hungwen Li  
IBM Almaden Research Center  
San Jose, California

ABSTRACT

Massively parallel computers are implemented by means of modules at different packaging levels. This paper discusses a hierarchical node clustering scheme (HNC) for packaging a class of reconfigurable processor arrays called Polymorphic Processor Array (PPA) which uses circuit-switching-based routers at each node to deliver a different topology at every instruction. The PPA family suffers from an unknown signal delay between arbitrary two nodes connected by the circuit-switched paths. This either forces the hardware clock to compromise to the worst signal delay or makes the software dependent on the system size. The use of the HNC scheme allows to obtain communication speed-up and automatic control, at the compiler level, over signal propagation delay.

1. INTRODUCTION

A Polymorphic Processor Array (PPA) [1] is a two-dimensional mesh connected computer, in which each node is equipped with a switch able to interconnect its four NEWS ports. PPA changes the switch setting, as part of the instruction, to speed up the data exchange between nodes; it shortens the distance between the nodes that have to communicate by short-circuiting all the intermediate nodes between the source and destination nodes.

PPA belongs to the class of reconfigurable processor arrays, such as the Mesh with Reconfigurable Bus (MRB) [2], the Processor Array with Reconfigurable Bus Systems (PARBS) [3], the Gated Connection Network (GCN) [4] and the Polymorphic-Torus [5]. The characteristic that distinguishes PPA from MRB, PARBS and GCN is that of providing only a limited set of reconfiguration capabilities: PPA allows the short-circuiting to occur only along rows and columns, whereas it does not support diagonal or arbitrary shape paths. At the first glance such a restriction seems to limit the power of PPA, however, as to be explained in the paper, it leads to a more programmable architecture,
thanks to the possibility of implementing effective hardware solutions, namely Hierarchical Node Clustering, supporting the data transmission along the short-circuited paths.

In PPA, like in MRB, PARBS and GCN, it is assumed that communication between any pair of nodes take an amount of time independent of the distance between the two nodes, which is equivalent to hypothesize that both the links and the switches are ideal; this is usually referred to as a "computation model". In reality, it is evident that in a physical system the propagation delay between two nodes is a function of the distance between them and besides, considering that reconfigurable processor arrays are proposed for modeling massively parallel computers, the worst case distance between two nodes may be very high. The gap between the idealized model and the implementation makes it impossible to write programs featuring the same complexity as the corresponding algorithms, unless proper design techniques are used to implement the basic operations of the idealized model.

In this paper we propose a design technique called Hierarchical Node Clustering (HNC) that reduces the maximum propagation delay in PPA from linear to logarithmic. The HNC scheme is based on the observation that the implementation of a massively parallel computer consists of a hierarchy of packages, each of which is characterized by a different propagation delay. Such a hierarchy can be exploited in the implementation of massively parallel computers based on hierarchical topologies such as tree, pyramid and hypercube. For example, considering that the logn steps of a scan operation [6] in an n-processor tree or hypercube exhibit a different degree of locality, it is possible to take advantage of such a locality and allow each step to be executed within the time actually needed instead of compromising to the longest delay required by the final reduction. On the contrary non hierarchical topologies such as ring and mesh always incur worst signal delay because any communication involves all the wires at all packaging levels. This paper shows how a non-hierarchical topology, such as PPA, can be benefited from the HNC scheme.

The paper is organized as follows. In the next section we introduce the PPA computation model, and in section 3 we describe how such a model can be implemented by taking advantage of the HNC technique. In section 4 we analyze the propagation delay in PPA both without HNC and with HNC, and in section 5 we discuss the cost and the benefit of HNC. In section 6 we provide some concluding remarks.

2. POLYMORPHIC PROCESSOR ARRAY COMPUTATION MODEL

A Polymorphic Processor Array (PPA), whose logical architecture is shown in Fig. 1, consists of a stack of three planes: the processor plane P, the memory plane M and the switch plane S. The P-plane is a two-dimensional array of processing elements (PEij) able to perform arithmetic and logical computations, the M-plane is a two-dimensional
array of random access memories \((M_{ij})\), and the S-plane is an array of switch-boxes \((S_{ij})\) interconnected by a two-dimensional torus, where \(i,j = 0, 1, \ldots, n-1\). Each set \((PE_{ij}, M_{ij}, S_{ij})\) is referred to as a PPA node and is called \(N_{ij}\). PPA is a SIMD architecture. Similar to other SIMD architectures, a central program controller broadcasts instructions and data to the P-plane and the memory addresses to the M-plane. However, unlike other SIMD architectures, the PPA program controller also broadcasts a switch instruction to the S-plane for the control of data communication.

A switch-box \(S_{ij}\), shown in Fig. 2, consists of an ideal switch (no delay) which can be oriented toward any of the four mesh directions (we shall refer to the orientation of the switch as the switch-box orientation) and which can be opened or shorted under program control (we shall refer to the OPEN/SHORT configuration of the switch as the switch-box configuration). The PPA computation model can be completely specified by the orientation and configuration of the switch-box.

The switch-box orientation, which can be N, E, W or S, is programmed by the central program controller through the switch instruction, therefore is identical for all
switch-boxes. On the contrary, the switch-box configuration, which can be OPEN or SHORT, depends on local data and can be different in each switch-box. (For example, the statement "where (local_data == 3) configuration = OPEN else configuration = SHORT" sets the switch-box configuration to OPEN in the nodes in which the element of local_data is 3 and sets the switch-box configuration to SHORT in the other nodes).

The operation of each PPA node depends on both the orientation and the configuration of the corresponding switch-box. At each instruction the PEs corresponding to switch-boxes in the OPEN configuration output a datum toward the current orientation, while the PEs corresponding to switch-boxes in the SHORT configuration let the data pass through them along the current orientation. Independent of the configuration, all the PEs load the datum incoming from the direction opposite to the current orientation (E and W are opposite directions and N and S are opposite directions).
Fig. 3 - Example of PPA data movement toward the E direction

Referring to the example in Fig. 3, in which the switch-box orientation is E, each node, independent of the switch-box configuration, receives a message from the W port. Depending on the switch-box configuration, in each node either the W port is connected to the E port (nodes a and b), or the processing element itself is connected to the E port (nodes c and d); nodes c and d also send a datum toward the E direction.

As described, the PPA distinguishes itself from the other computation models currently proposed for reconfigurable processor arrays in the following aspects:

1) *Directionality*: The PPA switch-box implements directional connections from one specific port to another, as opposed to MRB [2] and PARBS [3], that support bidirectional communication.

2) *One-dimensional interconnection*: The PPA switch-box allows the propagation of data either along the vertical direction (N->S or S->N) or along the horizontal direction (E->W or W->E), but not simultaneously. This is in contrast with GCN [4], MRB [2] and PARBS [PARBS], where the interconnection between non-opposite ports (N->E, N->W, S->E, S->W, E->N, E->S, W->N and W->S) is also supported.

These two important properties are the necessary conditions for the implementation of HNC, to be described next.

3. HIERARCHICAL NODE CLUSTERING IN PPA IMPLEMENTATION

A PPA system of $n^2$ nodes can be implemented by $p$ different packaging levels (for example the chip level, the multi-chip level and the board level), each of which is characterized by a propagation delay. We recursively define a module at packaging level $i+1$ as a two-dimensional array of $n_i \times n_i$ modules at packaging level $i$. In general $n_i^2$ modules at packaging level $i$ are contained in each module at packaging level $i+1$. 

\( i = 0, 1, \ldots , p - 2 \) and \( n^2 = \prod_{i=0}^{p-1} n_i \) is the total number of nodes of the PPA system. A module at packaging level 0 corresponds to a PPA node. Fig. 4 shows an example in which a module at packaging level 1 (e.g. chip) contains \( 4 \times 4 \) \( (n_0=4) \) nodes, a level 2 module (e.g. multi-chip carrier) contains \( 2 \times 2 \) chips, and a level 3 module (e.g. board) contains \( 3 \times 3 \) carriers.

The HNC scheme consists of adding a switch-box to each PPA module at each packaging level; under proper control, such a switch-box is able to bypass the module where the switch-box resides. As a result, the signal that, without HNC, would have to ripple through the nodes with SHORT configuration can be redirected via the module switch-box directly as shown in Fig. 5. Consequently, the propagation delay in a module at level \( i \) is reduced from the sum of the delays of the modules at level \( i - 1 \) to the delay of the module switch-box at level \( i \). in a module.

In the following we show the design of the module switch-box; we first give the switch-box design of the lowest packaging level, then generalize the design for the switch-boxes at the higher levels. Due to the PPA characteristics of directionality and one-dimensional interconnection introduced in section 2, the switch-box design can be considered for horizontal and vertical communication separately. From now on we will focus on a single PPA row, without loss of generality. Following the rule, the \( n \) nodes of a PPA row are hierarchically packaged in such a way that \( n_0 \) nodes are contained in each module at level 1, \( n_0 n_1 \) nodes are contained in each module at level 2, and in general \( \prod_{j=0}^{i-1} n_j \) nodes are contained in each module at level \( i \), \( i = 1, 2, \ldots , p - 1 \). The total number of nodes in the row is \( n = \prod_{i=0}^{p-1} n_i \).

Fig. 6 illustrates the switch-box design for the lowest level module. The figure is self-explanatory on the decoding of the orientation and the configuration. It also contains the bypassing circuit consisting of eight tri-state gates. The characteristic of directionality of the PPA switch-boxes allows for the use of electrically active elements, i.e. tri-state gates, instead of passive elements, such as pass-transistors, transmission-gates, and pre-charge/pull-down circuits like in [4] to implement the PPA switch-boxes. The benefits of tri-state implementation include the known fixed delay of each gate and the automatic regeneration of the signal, both of which support scalability.

A switch-box at generic packaging level \( i \), \( i = 1, \ldots , p - 1 \), shown in Fig. 7, has

1) \( 4 \prod_{k=0}^{i-1} n_k \) nearest-neighbor ports \( N_j, E_j, W_j, S_j, j = 0, 1, \ldots , \prod_{k=0}^{i-1} n_k - 1 \),

2) \( \prod_{k=0}^{i-2} n_k n_{i-1}^2 \) input control signals \( C^{i-1 \l_m}, \ l, m = 0, 1, \ldots , \prod_{k=0}^{i-2} n_k n_{i-1}^2 - 1 \) corresponding to the configuration of the modules at packaging level \( i - 1 \) included in it.
p-1 = 2, n_0 = 4, n_1 = 2, n_2 = 3, n = 24

Fig. 4 - Mapping PPA onto a multiple package implementation.
Submodule at level $i-1$

All the paths are $\prod_{k=0}^{i-1} n_k$ bits wide

Fig. 5 - Structure of a module at generic packaging level $i$
Fig. 6 - Node switch-box
3) $\prod_{k=0}^{i-1} n_k$ output status signals $C^i_j, j = 0, 1, \ldots, \prod_{k=0}^{i-1} n_k - 1$, corresponding to its configuration.

Considering that a switch-box associated with a module at packaging level $i$ bypasses a row or a column of modules at packaging level $i-1$ contained in it only when all the switch-boxes of such modules are in the SHORT configuration, it is possible to derive the following rules that give the $C^i$'s as functions of the $C^{i-1}$'s.

- case 1: orientation is E or W

$$C^1_h = \prod_{k=0}^{n_e-1} C_{hk}, \ h = 0, 1, \ldots, n_0 - 1$$

$$C^i_h = \prod_{k=0}^{n_e-1} \left[ C^{i-1}_{h \mod \prod_{m=0}^{i-1} n_m} \left| \begin{array}{c} h \mod \prod_{m=0}^{i-1} n_m \\ \prod_{m=0}^{i-1} n_m \end{array} \right. \right], \ h = 0, 1, \ldots, \prod_{m=0}^{i-1} n_m - 1$$

- case 2: orientation is N or S

$$C^1_h = \prod_{k=0}^{n_n-1} C_{hk}, \ h = 0, 1, \ldots, n_0 - 1$$

$$C^i_h = \prod_{k=0}^{n_n-1} \left[ C^{i-1}_{h \mod \prod_{m=0}^{i-1} n_m} \left| \begin{array}{c} h \mod \prod_{m=0}^{i-1} n_m \\ \prod_{m=0}^{i-1} n_m \end{array} \right. \right], \ h = 0, 1, \ldots, \prod_{m=0}^{i-1} n_m - 1$$

4. PROPAGATION DELAY ANALYSIS

In this section we introduce a propagation delay model for PPA implementation, that takes into account the use of modules at different packaging levels. We perform the analyses for PPA systems with and without HNC. Such analyses are important because the delay information can be used by the compiler to control the scheduling of the instructions. More specifically, in a real system a single clock period may be not long enough to perform the required communication, therefore it becomes important to know exactly how many clock periods are required. Knowing the exact delay avoids blindly adopting the worst case delay.

\[\text{1 The convention followed here is that "1" denotes a SHORT configuration and "0" denotes a OPEN configuration.}\]
\[ \text{STON}^i_h = C^i_h \; \text{and. (orientation } = N \text{)} \]
\[ \text{NTOS}^i_h = C^i_h \; \text{and. (orientation } = S \text{)} \]
\[ \text{ETOW}^i_h = C^i_h \; \text{and. (orientation } = W \text{)} \]
\[ \text{WTOE}^i_h = C^i_h \; \text{and. (orientation } = E \text{)} \]

Fig. 7 - Switch-box at generic packaging level \( i \).

In our definition, the PPA propagation delay model corresponds to a symmetric matrix \( T \) of size \( n \times n \), each element of which, \( T_{ud} \), corresponds to the propagation delay between node \( N_u \) and and node \( N_d \), as well as between node \( N_{ui} \) and node \( N_{di} \), \( i = 0, 1, \ldots, n-1 \). The following two analyses adopt this notation.

4.1. Propagation Delay without HNC

Let \( t^{\text{link}}_i, i = 0, 1, \ldots, p-1 \) be the propagation delay along the interconnection link between two neighbor modules at packaging level \( i \) and let \( t^{\text{switch}}_i, i = 0, 1, \ldots, p-1 \) be
the propagation delay of a module switch-box at packaging level $i$. The propagation delay between two generic nodes $s$ and $d$ $(1 \leq s \leq n-2, 2 \leq d \leq n-1, d > s)$ is given by

$$T_{sd} = (d - s) t_{\text{link}} 0 + (d - s - 2) t_{\text{switch}} 0 + \sum_{l=0}^{i-1} \alpha_l(s, d)(t_{\text{link}}_{i+1} - t_{\text{link}}_i)$$  \hspace{1cm} (1)$$

where

$$\alpha_l(s, d) = \text{pos} \left( \left\lfloor \frac{d}{\prod_{j=0}^{l} n_j} \right\rfloor - \left\lfloor \frac{s}{\prod_{j=0}^{l} n_j} \right\rfloor + 1 \right)$$

with

$$
p(x) = x \quad \text{if} \quad x \geq 0$$

$$
p(x) = 0 \quad \text{if} \quad x < 0
$$

The first two terms of equation (1) refer to the delay between the source and the destination assuming that only one packaging level (i.e. level 0) is implemented. When a new packaging level $i+1$ is introduced, the propagation delays $t_{\text{link}}_i$ along the links that interconnect modules at packaging level $i+1$ must be replaced by $t_{\text{link}}_{i+1}$. The term $\alpha_l(s, d)$ represents the number of boundaries of modules at packaging level $i$ located between nodes $s$ and $d$. An example of this calculation is shown in Fig. 8a.

4.2. Propagation Delay with HNC

The propagation delay between two generic nodes $s$ and $d$ is given by

$$T_{\text{HNC}}^{sd} = T_{sd} + \sum_{i=0}^{n-1} \beta_i(s, d)(t_{\text{switch}}_{i+1} - n_i) t_{\text{switch}}_i -(n_i - 1)t_{\text{link}}_i$$ \hspace{1cm} (2)$$

where

$$\beta_i(s, d) = \text{pos} \left( \left\lfloor \frac{d}{\prod_{j=0}^{i-1} n_j} \right\rfloor - \left\lfloor \frac{s}{\prod_{j=0}^{i-1} n_j} \right\rfloor \right)$$

and $\alpha_l(s, d)$ and $\text{pos}(x)$ are defined as above.

The delay $T_{\text{HNC}}^{sd}$ is derived by taking $T_{sd}$ as a starting point. When a module at level $i+1$ is bypassed the following corrections to $T_{sd}$ are required:

1) the delay of the module switch-box at level $i+1$ (i.e. $t_{\text{switch}}_{i+1}$) is added;

2) the sum of the delays of the switch-boxes at level $i$ (i.e. $t_{\text{switch}}_i$) is subtracted;

3) the sum of the delays of the links at level $i$ (i.e. $t_{\text{link}}_i$) is subtracted.

The term $\beta_i(s, d)$ represents the number of modules at packaging level $i$ entirely contained between nodes $s$ and $d$. An example of this calculation is shown in Fig. 8b.
\[ T_{2,17} = 15t_0 + 14t_0 + 4t_1 - 4t_0 + 2t_2 - 2t_1 = \]
\[ = 11t_0 + 14t_0 + 2t_1 + 2t_1 \]

a)

\[ T_{2,17}^{\text{HNC}} = T_{2,17} + 3t_1 - 12t_0 - 9t_0 + t_2 - 2t_1 - t_1 = \]
\[ = 2t_0 + 2t_0 + t_1 + t_1 + 2t_2 + t_2 \]

b)

Fig. 8 - Example of Computation of the Propagation Delay.
   a) without HNC, b) with HNC.
5. SIMULATION RESULTS AND DISCUSSION

5.1. Communication Speed-Up

The most evident benefit of the HNC technique is the reduction of the propagation delay between any pairs of PPA nodes. While in a PPA system without HNC, the propagation delay between two nodes is a linear function of their distance, in a PPA system adopting HNC the propagation delay between two nodes can be reduced up to a logarithmic function of their distance. Such a limit, shown in Fig. 9, is reached when each cluster at level \( i \), \( 0 < i < p \) contains two clusters at level \( i-1 \) and the propagation delay is the same at every packaging level.

The logarithmic reduction occurs only in an idealized situation. In reality, a larger size of the clusters is supported by the packaging technology. Fig. 10 illustrates the propagation delay in a realistic case. In such a case we have assumed we have only two packaging levels, namely the integrated-circuit level (IC) and the printed-circuit-board (PCB) level, both characterized by switch-box propagation delay and interconnection link propagation delay of \( 4nsec \). In each IC there reside \( 16 \times 16 \) nodes and in each PCB we package \( 16 \times 16 \) IC's; this constructs a processor array of \( 256 \times 256 = 64K \) nodes. In Fig. 10 we compare the curves of the propagation delays without HNC (curve (1)) and with HNC (curve (2)). Due to the fact that HNC can only bypass IC's, the effect of HNC on the delay curve is only a reduction of the slope, instead of a trasformation from a linear function to a logarithmic function. Further improvement can be made toward logarithmic reduction by placing additional switch-boxes, at the board level, that bypass groups of short-circuited chips. Curve (3) shows how the propagation delay is improved by placing bypass switch-boxes every two, four and eight chips. Fig. 10 shows how HNC helps to reduce the propagation delay of a PPA system in a controllable and quantizable manner. Depending on the maximum acceptable propagation delay, a proper number of clustering levels can be chosen to guarantee the worst delay and the delay for certain desired communication topologies. The upper bound of the propagation delay is limited by the logarithmic behaviour shown in Fig. 9 where every pair of clusters at any level is grouped into a cluster at the upper level.

5.2. Scalability

Scalability is always an important issue for the design of a parallel system. For reconfigurable processor array, the scalability hampers both the hardware and the software. Without the insight and the control on the delay time, the system clock depends on the system size. Similarly, the program depends on the system size and needs to be re-written when the size changes. Such a system design is not scalable and is a poor model for programmers.
curve (1): no HNC

curve (2): $p = 8$, $n_i = 2$, $t^\text{link}_{i} = t^\text{switch}_{i} = 1$ logical time unit, $i = 0, 1, \ldots, p-1$

curve (3): reference logarithmic curve ($3 \log(d-s)$)

Fig. 9 - Propagation delay (in logical time units) versus distance: ideal model
curve (1): no HNC

curve (2): a) \( p = 2, n_0 = 16, n_1 = 16, t^{\text{link}}_0 = t^{\text{switch}}_0 = 4\text{nsec} \), \( t^{\text{link}}_1 = t^{\text{switch}}_1 = 4\text{nsec} \)

curve (3): \( p = 5, n_0 = 16, n_1 = n_2 = n_3 = n_4 = 2 \)
\( t^{\text{link}}_0 = t^{\text{link}}_1 = t^{\text{link}}_2 = t^{\text{link}}_3 = t^{\text{link}}_4 = 4\text{nsec} \)
\( t^{\text{switch}}_0 = t^{\text{switch}}_1 = t^{\text{switch}}_2 = t^{\text{switch}}_3 = t^{\text{switch}}_4 = 4\text{nsec} \)

Fig. 10 - Propagation delay (in nanoseconds) versus distance in a realistic case
The difficulty to maintain scalability is not unique to PPA. For a bus-based multiprocessor system, a protocol and complex hardware are needed to support scalability. Even so, the bus-based system usually fails quickly and exhibits poor performance. For a parallel computer based on high-degree network such as a hypercube, the difficulty of scalability lies on the linear growing of the wires. Even for a sequential system, when a large amount of memory is built for the system, the memory access time becomes "unscaleable", consequently complex cache system is popularly adopted to maintain the scalability.

An interesting observation here is that when "size" grows, the curse due to "size" damages the scalability from different aspects: memory access time for a sequential processor, wiring complexity for a high-degree network, or for reconfigurable processor array the undeterministic delay time. The hierarchical node clustering scheme is a solution to remedy the "unscaleable" aspect of PPA.

5.3. Impact on programming

With the experience in developing massively parallel processing hardware in the past decade, experts in the field have gained enough confidence that one can build a large massively parallel hardware with controlled and careful engineering. However, it is a challenge to develop system software and programs for a massively parallel computer. As a result, hardware architect has less hesitation to either include or exclude hardware features only if it helps to ease the software development.

As discussed in the previous sections, the HNC scheme provides a controlled insight into the signal delay in a very large PPA. A very immediate benefit of HNC is that it shortens the signal delay hence the system clock if the architect adopts the worst-case delay for choosing the clock. What is more important is that, with the timing information provided by HNC, one can choose a clock cycle shorter than the worst-case delay.

Specifically, for position-dependent reconfiguration, the delay is known at compile time. It is therefore possible for the compiler to allocate enough, but just enough, time to accommodate the delay. This removes the burden of the worst case for a reconfigurable system in general. Similarly, for regular configurations such as tree and pyramid (which uses 2's power distances for reconfiguration), the delay is usually pre-calculated and stored in compiler tables. Only for the data-dependent reconfiguration, the delay is not known at the compile time and therefore the maximum delay needs to be used.

The allowance of code optimization is another benefit offered by HNC. Since in PPA computation and communication can be conducted independently, if the communication delay is known at the compile time, it is possible to pack other computations that are not dependent on the on-going communication. This is similar to the code optimization for Superscalar in which multiple instructions are issued and allowance for code
compaction exists when there is an independence between consecutive instructions.

We reminded the reader in the previous sections that there are many models for reconfigurable processor array and the model discussed in this paper is not the most powerful one considering that it only supports communication along the row or column direction. However, it is this restriction that supports the HNC scheme, which possesses nice properties of known delay at compile time. Thus, the programmer is released from the timing burden of a reconfigurable system with the help of a compiler. Furthermore, there exist chances for code optimization. From the software viewpoint, a simpler and less powerful model becomes a better design solution for a massively parallel system.

6. CONCLUDING REMARKS

In this paper we have analyzed the propagation delay in reconfigurable processor arrays. While in reconfigurable processor array computation models communication between arbitrary nodes is assumed to be instantaneous, it is not in implementations. The gap between computation models and implementations can be filled by the Hierarchical Node Clustering technique introduced in this paper, which allows the bypassing of clustered nodes of various sizes in long distance communication, and the management of the signal delay in a controlled manner.

The key benefit of narrowing the gap between a model and its implementation is the ease of programming. Using the HNC scheme makes PPA scalable; the signal delay time can be controlled and be more independent of the size of PPA hence PPA programs can be written subject to "the model" without concerning with "the implementation". With the significant reduction of the signal delay, the programming subject to the model is efficient and does not have to compromise with the worst case delay.

The application of HNC in PPA allows to significantly shorten the propagation delay between any pair of nodes. Coupling HNC with the possibility of overlapping computation and communication allows the compiler to implement compaction techniques, in such a way to optimize the code generated and to eliminate (or at least to reduce) the number of idle states to be inserted after instructions requiring long distance communication.

Not any reconfigurable processor array can take advantage of the proposed technique. HNC requires that communication be carried out only along one of the two dimensions at a time, namely either horizontally or vertically; the automatic spreading of data along arbitrary multi-shaped paths is not allowed. This reveals an interesting relationship between the "power of the model" and "programming". The "row/column only" PPA is a less powerful model that exhibits a limited number of reconfiguration capabilities. Nevertheless, with the HNC scheme, it becomes a scalable architecture and enjoys the programming efficiency.
The HNC technique contributes to the hardware aspect by reducing the signal delay; its most important value, however, lies in the profound impact on programming massively parallel computers.

References


