International Journal of Automation and Computing 9(3), June 2012, 280-287
DOI: 10.1007/s11633-012-0645-1

# A Partitioning Methodology That Optimizes the Communication Cost for Reconfigurable Computing Systems

Ramzi Ayadi[1] Bouraoui Ouni[1] Abdellatif Mtibaa[1, 2]

[1]Laboratory of Electronics and Microelectronics, Faculty of Science of Monastir, University of Monastir, Monastir 5019, Tunisia
[2]Department of Electrical Engineering, National Engineering School of Monastir, University of Monastir, Monastir 5019, Tunisia

**Abstract:** This paper focuses on the design process for reconfigurable architecture. Our contribution focuses on introducing a new temporal partitioning algorithm. Our algorithm is based on typical mathematic flow to solve the temporal partitioning problem. This algorithm optimizes the transfer of data required between design partitions and the reconfiguration overhead. Results show that our algorithm considerably decreases the communication cost and the latency compared with other well known algorithms.

**Keywords:** Temporal partitioning, data flow graph, communication cost, reconfigurable computing systems, field-programmable gate array (FPGA).

## 1 Introduction

The temporal partitioning problem[1−3] can be seen as a graph-based problem. A program or application can be modelled by a data flow graph. Then, the temporal partitioning divides the input graph into temporal partitions that are configured one after another on the reconfigurable device[4, 5]. Each partition is also called a stage or a micro-cycle, and all the micro-cycles form one user cycle. The first temporal partition receives input data, performs computations and stores intermediate data into on-board memory. The device is then reconfigured for the next segment, which computes results based on the intermediate data, from the previous partitions. Fig. 1 shows a part of a design that has been partitioned into four partitions. Assuming that a node requires a configurable logic block (CLB) and each arc has a 1-byte width, and also assuming that there is a device with a size of 4 CLB and a memory with 3 bytes is available for communication. The partitioning, shown in Fig. 1 (a), needs five CLBs and five bytes while that shown in Fig. 1 (b) uses only three CLBs and three bytes. Therefore, the partitioning, shown in Fig. 1 (a), is undesirable.

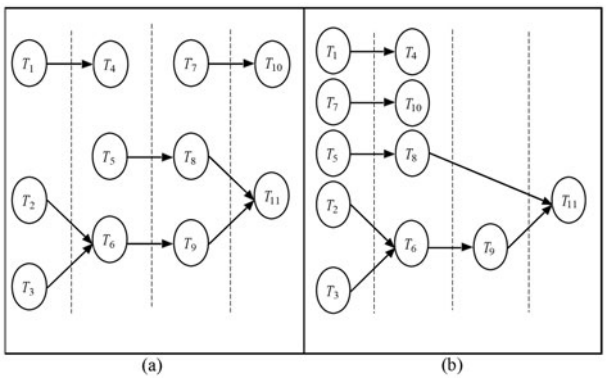


Fig. 1 Temporal partitioning

Manuscript received December 21, 2010; revised September 7, 2011

The main objective of most of the related works is either to find the minimal size of reconfigurable area to accomplish the graph within a fixed limit of time or to find the minimal execution time of the input graph on a fixed-size of area[6−9]. However, the proposed approach focuses on minimizing the required data transfer between different temporal partitions of the design. The goal to be reached during partitioning is the minimization of the communication overhead among the partitions, which also means the minimization of the use of memory. In Fig. 2, we present an illustrative example, without taking care of any constraint, of two temporal partitionings. In Fig. 2 (a), the total communication cost across partitions is 35. However, the sum of the communication cost in Fig. 2 (b) is 19. Hence, to reduce the communication cost, we favor the second algorithm.

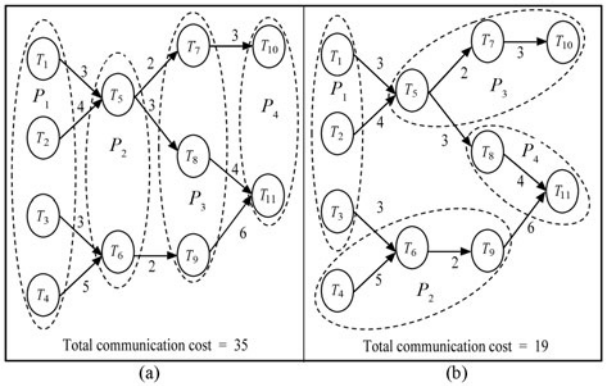


Fig. 2 Total communication cost

## 2 Related work

In the literature, often the network flow algorithm has been used to reduce the communication cost across temporal partitions. The first network flow algorithms has been used in [10−12] and improved in [13]. The method is a recursive bipartition approach that successively partitions

a set of remaining nodes in two sets, one of which is a final partition, whereas a further partition step must be applied on the second one. The initial network algorithm is shown below as presented in [10, 11].

**Begin**

**Step 1.** Construct graph $G'$ from graph $G$ by net modelling.

**Step 2.** Pick a pair of node $s$ and $t$ in $G'$ as source and sink.

**Step 3.** Find a min cut $C$ in $G'$. Let $X$ be the sub-graph reachable from $s$ through augmenting path, and $X'$ be the rest.

**Step 4.** If $(Lr \leqslant w(X) \leqslant Ur)$ then stop and return $C$ as solution.

**Step 5.** If $(w(X) < Lr)$ then collapse all nodes in $X$ to $S$ pick a node $v$ in $X'$, and collapse $v$ to $s$ go to Step 3.

**Step 6.** If $(w(X) > Ur)$ then collapse all nodes in $X'$ to $t$ pick a node $v$ in $X$, and collapse $v$ to $t$ go to Step 3.

**End**

$w(X)$ is the total area of all nodes in $X$; $Lr = (1-\varepsilon)R_{\max}$, $R_{\max}$ is the area of the device; $Ur = (1-\varepsilon)R_{\max}$; $\varepsilon = 0.05$; $S$ is the source node; $t$ is the sink node. Let us consider the graph $G$ of Fig. 3.

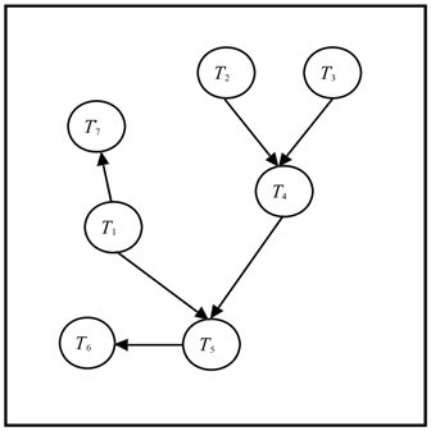


Fig. 3 Graph $G$

Let us assume 200 CLBs be the area of the device, 100 CLBs be the area of the multiplier, and 50 CLBs be the area of the adder, the comparator and the multiplexer. And let us assume a memory with 50 bytes available for communication and each edge has a 32-bit width. We applied the network flow algorithm on the graph of Fig. 3. The result is shown in Fig. 4, the network flow algorithm puts nodes $T_2$, $T_3$, and $T_4$ in partition $P_1$, nodes $T_1$, $T_5$, and $T_6$ in partition $P_2$ and node $T_7$ in partition $P_3$.

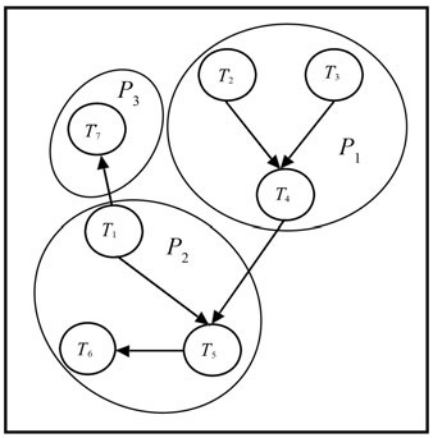


Fig. 4 Temporal partitioning

The network flow may minimize the communication cost. However, the model is constructed by inserting a great amount of nodes and edges in the original graph. The resulting graph may grow too big. In the worst case, the number of nodes in the new graph can be twice the number of the nodes in the original graph. The number of additional edges also grows dramatically and becomes difficult to handle. Further, the network flow algorithm, as shown above, is a heuristic algorithm, and in fact there is no a mathematical model behind it.

## 3 Data flow graph

A data flow graph (DFG) is a directed acyclic graph $G = (V, E)$, where $V$ is a set of nodes $|V| = n$ and $E$ is a set of edges. A directed edge $e_{ij} \in E$ represents the dependence between nodes $(T_i, T_j)$. For each edge $e_{ij}$, there is a weight $a_{ij}$ that represents the communication cost between node $T_i$ and node $T_j$. We assume that each node has an equivalent hardware implementation, which occupies an area on the chip. Therefore, the nodes as well as the edges in a DFG have some characteristics such as area, latency and width that are derived from the hardware resources used later to implement nodes.

### 3.1 Node and edge parameters

Given a node $T_i \in V$ and $e_{ij} \in E$.

1) $a_i$ denotes the area of $T_i$.

2) The latency $L_i$ is the time needed to execute $T_i$.

3) For a given edge $e_{ij}$ which defines a data dependency between $T_i$ and $T_j$, we define the weight $\alpha_{i,j}$ of $e_{ij}$ as the amount of data transferred from $T_i$ to $T_j$.

## 4 Temporal partitioning

A temporal partitioning $P$ of the graph $G = (V, E)$, is its division into some disjoint partitions such that $P = \{P_1 \cdots P_k\}$. A temporal partitioning is feasible in accordance to a reconfigurable device $H$ with area $A(H)$ and pins $T(H)$ (number of programmable input/outputs (I/Os) per device); if the two conditions are verified:

$$\forall P_i \in P;\ A(P_i) \leqslant A(H)$$

$$TCCost = \sum_{i=1}^{K} CCost(P_m) = \sum_{m=1}^{K} \sum_{T_i \in P_m; T_j \in \overline{P}_m} \alpha_{i,j} \leqslant T(H)$$

where $TCCost$ denotes total communication cost, and $CCost$ denotes communication cost.

In the rest of this section, we are interested in the explanation of these two conditions.

Given a temporal partitioning $P = \{P_1 \cdots P_k\}$ of the data flow graph $G = (V, E)$, the area constraint is satisfied if and only if:

$$\forall P_i \in P;\ A(P_i) \leqslant A(H) \quad (1)$$

where $A(P_i)$ is the area of partition $P_i$; $A(H)$ is the area of the device. Or the area of partition $P_i$ equals the area of nodes belong to partition $P_i \Rightarrow$

$$A(P_i) = \sum_{T_i \in P_i} a_i. \tag{2}$$

Based on (1) and (2) to satisfy the area constraint

$$\forall P_i \in P; \sum_{T_i \in P_i} a_i \leqslant A(H). \tag{3}$$

Given a temporal partitioning $P = \{P_1 \cdots P_k\}$ of the data flow graph $G = (V, E)$, the pins constraint is satisfied if and only if:

$$TCCost = \sum_{i=1}^{K} CCost(P_m) = \sum_{m=1}^{K} \sum_{T_i \in P_m; T_j \in \overline{P}_m} a_{i,j} \leqslant T(H). \tag{4}$$

In fact, if the variable $a_{i,j} \neq 0$ signifies that $T_j$ depends on $T_i$. When node $T_i$ is being placed in partition $P_m$ and $T_j$ is being placed outside $P_m$, then the data being communicated between them will have to be stored in the memory. Consequently, the sum of all the data being communicated across all partition should be less than the pins constraint.

## 5 Proposed algorithm

Our algorithm aims to solve the following problem: Given a DFG $G = (V, E)$ and a set of constraints: Find the way of graph partitioning with optimal number of temporal partitions such that the communication cost has the lowest value while respecting all constraints.

Our algorithm is composed of two main steps. The first step aims to find an initial partitioning $P_{in}$ of the graph. This step gives the optimal solution in term of communication cost. Next, if the area constraint is satisfied after the first step then we adopt the initial partitioning, else we go to the second step. Hence, the second step aims to find the final partitioning $P$ of the graph while satisfying the area constraint. If the second step cannot find a feasible scheduling then we relax the number of partitions by one and the algorithm goes to the first step. And, we restart to find a feasible solution with the new number of partitions.

### 5.1 First step: initial partitioning

Given a data flow graph $G = (V, E)$, we define:
The weighted adjacency matrix $W$ as follows:

$$W_{i,j} = a_{i,j} = 0.$$

The degree matrix $D$ as follow:

$$D_{i,j} = \sum_{i=1}^{n} W_{i,j}, \quad D_{i,i} = 0.$$

The Laplacian of $G$ as follow:

$$L = D - W.$$

**Lemma 1.** The matrix $L$ satisfies the following properties[14]:

1) For every vector $X = \{x_1, x_2, \cdots, x_n\} \in \mathbf{R}^n$; we have

$$X^{\mathrm{T}} L X = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} W_{i,j} (x_i - x_j)^2.$$

2) $L$ is symmetric and positive semi-definite.

3) $L$ has $n$ non-negative, real-valued eigenvalues $0 = \lambda_1 \leqslant \lambda_2 \leqslant \cdots \leqslant \lambda_n$.

**How to minimize the communication cost?**

Our main goal is to find the way of partitioning the graph such as the communication cost has lowest value. This section shows how to achieve a good solution to such graph partitioning problems.

Given a temporal partitioning of $G = (E, V)$ into $k$ disjoint partitions $P = \{P_1, P_2, \cdots, P_k\}$; the communication cost, $CCost(P_m)$, of partition $P_m$ has been defined in [15] as follows:

$$CCost(P_m) = \frac{1}{2} \left( \frac{\sum_{T_i \in P_m; T_j \in |\overline{P_m}|} W_{i,j}}{|P_m|} \right). \tag{5}$$

This implies that

$$TCCost = \sum_{i=1}^{K} CCost(P_m) = \frac{|V|}{2} \sum_{i=1}^{k} \frac{\sum_{T_i \in P_m; T_j \in |\overline{P_m}|} W_{i,j}}{|P_m||\overline{P_m}|} \tag{6}$$

where $TCCost$ is the total communication cost, $|P_m|$ is the number of nodes inside partition $P_m$, $|\overline{P_m}|$ is the number of nodes outside the partition $P_m$. Hence, we have $|P_m| + |\overline{P_m}| = |V| = n$.

**Lemma 2.** Given an indicator vector $X_m$ defined as follow: $X_m(i) = \{X_m(1), X_m(2), \cdots, X_m(n)\}^{\mathrm{T}}$, where $m = 1, 2, \cdots, k$ and $i = 1, 2, \cdots, n$ are defined as

$$X_m(i) = \begin{cases} \dfrac{1}{\sqrt{|\overline{P_m}|}}, & \text{if } T_i \in P_m \\ 0, & \text{otherwise} \end{cases}$$

we have

$$TCCost = \sum_{i=1}^{K} X_m^{\mathrm{T}} L X_m. \tag{7}$$

We introduce a matrix $Xp$ $(n \times k)$ that contains the $k$ indicator vectors as columns.

We can check that

$$X_m^{\mathrm{T}} L X_m = (X_p^{\mathrm{T}} L X_p)_{m,\,m}. \tag{8}$$

Overall, we can achieve

$$TCCost = \sum_{m=1}^{K} X_m^{\mathrm{T}} L X_m = \sum_{m=1}^{K} (X_p^{\mathrm{T}} L X_p)_{m,m} = \mathrm{tr}(X_p^{\mathrm{T}} L X_p)_{m,m}. \tag{9}$$

Therefore, using (5), the problem of communication cost minimization can be expressed as

$$\mathrm{Minimize}(T\_Com\_Cost) \rightarrow \mathrm{Minimize}(\mathrm{tr}(X_p^{\mathrm{T}} L X_p). \tag{10}$$

The standard form of a trace minimization problem can be solved by choosing $X_p$ as the matrix that contains the

first $k$ eigenvectors corresponding to the $k$ smallest eigenvalues of matrix $L$ as columns.

**Lemma 3.** Given a $(n \times n)$ matrix $M_p$ as follows:

$$M_{i,j} = \begin{cases} \frac{1}{|P_m|}, & \text{if } T_i \text{ , } T_j \in P_m \\ 0, & \text{otherwise} \end{cases}$$

we have $X_m^{\mathrm{T}} L X_m = M_p$.

The initial partitioning step of our algorithm is summarized by the following steps:

**Step 1.** Compute the minimum number of partitions $K = Min\_Part = \lceil Area(G)/Area(H) \rceil$.

**Step 2.** Compute the Laplacian matrix $L(G)$ of $G$.

**Step 3.** Compute $k$ lowest eigenvalues of $L(G)$.

**Step 4.** Construct the $(n \times k)$ matrix $X_p$ that have the $K$ eignvectors as columns.

**Step 5.** Compute $Z = X_p X_p^{\mathrm{T}}$.

**Step 6.** Construct the $(n \times n)$ matrix $M_p = M_{i,j}$ from $Z$. $M_{i,j} = 1$ if $Z_{i,j} \geqslant 1/n$, 0 otherwise.

**Step 7.** Generate the initial partitioning from matrix $M_p$.

**Step 8.** If the area constraint is satisfied then final partitioning = initial partitioning; else go to Step 2 (we mean by "go to Step 2": go to final partitioning step).

## 5.2 Second step: final partitioning

In this step, we start from the initial partitioning $P_{in}$ given by the first step and the set of partitions $P_i \in P_{in}$, where $A(P_i) > A(H)$. Our technique balances nodes from partition $P_i$ to $P_j$ or conversely until the satisfaction of the area constraint. The balance of nodes is based on the force $F(T_i,\ P_i \to P_j)$ associated with partition $P_i$ on a node $T_i$ to be scheduled into partition $P_j$ and on the force $F(T_i,\ P_j \to P_i)$ associated with partition $P_j$ on a node $T_i$ to be scheduled into partition $P_i$. For instance, let us assume that $P_i < P_j$; $P_i,\ P_j \in P_{in}$.

These forces are calculated as follow:

$$F(T_i, P_i \to P_j) = \delta_1(T_i) \times OF(T_i) \tag{11}$$

where $\delta_1(T_i) = 0$, if there is a node $T_j \in P_i$ and $T_j$ is an output of $T_i$, otherwise $\delta_1(T_i) = 1$.

$$OF(T_i) = (Nu(T_i) + 1). \tag{12}$$

Given two nodes $T_i$ and $T_j \in P_i$

$$Nu(T_i) = \sum_{T_j \in P_i} \beta_{i,\,j} T_j \tag{13}$$

where $\beta_{i,\,j} = 1$, if $T_j$ is an input of $T_i$, 0 otherwise

$$F(T_i, P_j \to P_i) = \delta_2(T_i) \times InF(T_i) \tag{14}$$

where $\delta_2(T_i) = 0$, if there is a node $T_j \in P_i$ and $T_j$ is an input of $T_i$, otherwise $\delta_2(T_i) = 1$.

$$InF(T_i) = (Nq(T_i) + 1). \tag{15}$$

Given two nodes $T_i$ and $T_j \in P_j$

$$Nq(T_i) = \sum_{T_j \in P_j} \phi_{i,j} T_j \tag{16}$$

where $\phi_{i,j} = 1$, if $T_j$ is an output of $T_i$, 0 otherwise.

In general, due to the scheduling of one node, other node schedules will also be affected. At each iteration, the force of every node being scheduled in every possible partition is computed. Then, the distribution graph is updated and the process repeats until no more nodes remain to be scheduled.

## 5.3 Example

Applying the steps of our algorithm on the graph of Fig. 5, let 800 CLB be the area of the device.

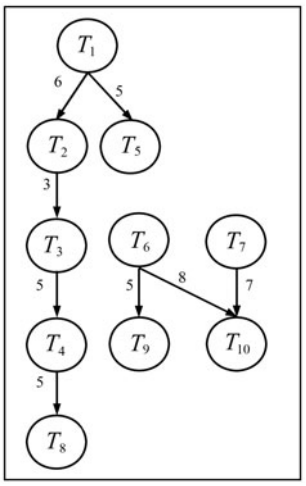


Fig. 5 Graph $G$

Table 1 Areas of nodes

| Nodes | Area (CLB) |
|---|---|
| $T_1$ | 225 |
| $T_2$ | 210 |
| $T_3$ | 220 |
| $T_4$ | 200 |
| $T_5$ | 276 |
| $T_6$ | 196 |
| $T_7$ | 210 |
| $T_8$ | 167 |
| $T_9$ | 200 |
| $T1_{10}$ | 230 |

**Step 1.** $k = \lceil Area(G)/Area(H) \rceil \lceil 2134/800 \rceil = 3$.

**Step 2.** Laplacian matrix $L(G)$ of $G$ is

$$\begin{vmatrix} 11 & -5 & 0 & 0 & -5 & 0 & 0 & 0 & 0 & 0 \\ -5 & 9 & -3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -3 & 7 & -4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -4 & 9 & 0 & 0 & 0 & -5 & 0 & 0 \\ -5 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 13 & 0 & 0 & -5 & -3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 7 & 0 & 0 & -7 \\ 0 & 0 & 0 & 5 & 0 & 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -5 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 & -8 & -7 & 0 & 0 & 15 \end{vmatrix}$$

**Step 3.1.** Eigenvalues of matrix $L(G)$ are

$\lambda_1 = 0 \leqslant \lambda_2 = 0 \leqslant \lambda_3 = 1.0671 \leqslant \lambda_4 = 3.899 \leqslant \lambda_5 = 4.9158 \leqslant \lambda_6 = 8.2945 \leqslant \lambda_7 = 11.8389 \leqslant \lambda_8 = 14.0258 \leqslant \lambda_9 = 17.6967 \leqslant \lambda_{10} = 24.2618$.

**Step 3.2.** Eigenvectors of matrix $L(G)$

$$\left|\begin{array}{ccccccc} 0.40 & 0.03 & 0.40 & 0 & 0 & -0.39 & 0 \\ 0.40 & 0.03 & 0.23 & 0 & -0.41 & -0.52 & 0 \\ 0.40 & 0.03 & -0.17 & 0 & -0.58 & 0.48 & 0 \\ 0.40 & 0.03 & -0.43 & 0 & 0 & 0.23 & 0 \\ 0.40 & 0.03 & 0.51 & 0 & 0.50 & 0.45 & 0 \\ 0.03 & -0.49 & 0 & -0.15 & 0 & 0 & 0.61 \\ 0.03 & -0.49 & 0 & 0.61 & 0 & 0 & -0.53 \\ 0.40 & 0.03 & -0.55 & 0 & 0.47 & -0.35 & 0 \\ 0.03 & -0.49 & 0 & -0.72 & 0 & 0 & -0.44 \\ 0.03 & -0.49 & 0 & 0.27 & 0 & 0 & 0.36 \end{array}\right.$$

$$\left.\begin{array}{ccc} -0.22 & -0.72 & 0 \\ 0 & 0.57 & 0 \\ 0.42 & -0.20 & 0 \\ -0.75 & 0.12 & 0 \\ 0.12 & 0.28 & 0 \\ 0 & 0 & -0.53 \\ 0.42 & 0 & -0.29 \\ 0 & -0.04 & 0 \\ 0 & 0 & 0.15 \\ 0 & 0 & 0.73 \end{array}\right|$$

**Step 4.** Matrix $X_p$, columns of $X_p$ = (Eigenvectors of the lowest eigenvalues) is

$$\left|\begin{array}{ccc} 0.40 & 0.03 & 0.40 \\ 0.40 & 0.03 & 0.40 \\ 0.40 & 0.03 & 0.23 \\ 0.40 & 0.03 & -0.17 \\ 0.40 & 0.03 & -0.43 \\ 0.40 & 0.03 & 0.51 \\ 0.03 & -0.49 & 0 \\ 0.03 & -0.49 & 0 \\ 0.40 & 0.03 & -0.55 \\ 0.03 & -0.49 & 0 \\ 0.03 & -0.49 & 0 \end{array}\right|$$

**Step 5.** Matrix $Z = X_p X_p^{\mathrm{T}}$ is

$$\left|\begin{array}{ccccccc} Nodes & T_1 & T_2 & T_3 & T_4 & T_5 & T_6 \\ T_1 & 0.32 & 0.26 & 0.09 & 0 & 0.37 & 0 \\ T_2 & 0.26 & 0.22 & -0.12 & 0.06 & 0.28 & 0 \\ T_3 & 0.09 & -0.12 & 0.19 & 0.24 & 0.07 & 0 \\ T_4 & 0 & 0.06 & 0.24 & 0.35 & -0.05 & 0 \\ T_5 & 0.37 & 0.28 & 0.07 & -0.05 & 0.42 & 0 \\ T_6 & 0 & 0 & 0 & 0 & 0 & 0.25 \\ T_7 & 0 & 0 & 0 & 0 & 0 & 0.25 \\ T_8 & -0.05 & 0.03 & 0.26 & 0.40 & -0.11 & 0 \\ T_9 & 0 & 0 & 0 & 0 & 0 & 0.25 \\ T_{10} & 0 & 0 & 0 & 0 & 0 & 0.25 \end{array}\right.$$

$$\left.\begin{array}{cccc} T_7 & T_8 & T_9 & T_{10} \\ 0 & -0.05 & 0 & 0 \\ 0 & 0.03 & 0 & 0 \\ 0 & 0.26 & 0 & 0 \\ 0 & 0.40 & 0 & 0 \\ 0 & -0.11 & 0 & 0 \\ 0.25 & 0 & 0.25 & 0.25 \\ 0.25 & 0 & 0.25 & 0.25 \\ 0 & 0.46 & 0 & 0 \\ 0.25 & 0 & 0.25 & 0.251 \\ 0.25 & 0 & 0.25 & 0.25 \end{array}\right|$$

**Step 6.** The matrix $M_p$

$$\left|\begin{array}{ccccccccccc} Nodes & T_1 & T_2 & T_3 & T_4 & T_5 & T_6 & T_7 & T_8 & T_9 & T_{10} \\ T_1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ T_2 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ T_3 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ T_4 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ T_5 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ T_6 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ T_7 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ T_8 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ T_9 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ T_{10} & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{array}\right|$$

To build the initial partitioning, we use the propriety of Lemma 3; if $M_{i,j} = M_{j,i} = 1$ then nodes $(T_i, T_j)$ belong to the same partition, else $(T_i, T_j)$ belong to deferent partitions.

**Step 7.** Initial partitioning $(P_{in})$, with $K = 3$.

Considering the partitioning solution of the graph shown above in shown in Fig. 6,

$$\begin{array}{ll} P_1 = T_1T_2T_5 & Area\ (P_1) = 736 \text{ CLBs} \\ P_3 = T_3T_4T_8 & Area\ (P_2) = 597 \text{ CLBs} \\ P_3 = T_6T_7T_9T_{10} & Area(P_3) = 836 \text{ CLBs}. \end{array}$$

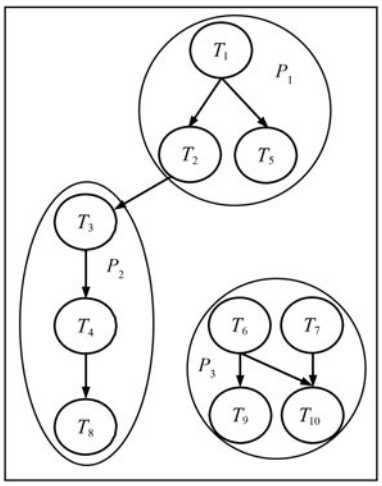


Fig. 6 Initial partitioning

The area constraint is not satisfied for partition $P_3$.

To find the final partitioning, first our algorithm compute the force associated to each node in partition 3 to be scheduled in partition 2.

$$\begin{array}{ll} F(T_6, P_3 \rightarrow P_2) = 0,66, & F(T_7, P_3 \rightarrow P_2) = 0,5 \\ F(T_9, P_3 \rightarrow P_2) = 0, & F(T_{10}, P_3 \rightarrow P_2) = 0. \end{array}$$

Next, the algorithm displaces node 7 from $P_3$ to $P_2$ since it has the lowest nonzero force, hence the new solution will be

$$\begin{array}{ll} P_1 = T_1T_2T_5 & Area(P_1) = 736 \text{ CLBs} \\ P_3 = T_3T_4T_8T_7 & Area(P_2) = 797 \text{ CLBs} \\ P_3 = T_6T_9T_{10} & Area(P_3) = 626 \text{ CLBs}. \end{array}$$

Finally, since the new partitioning, shown in Fig. 7, satisfies the area constraint, we adopt it as final partitioning.

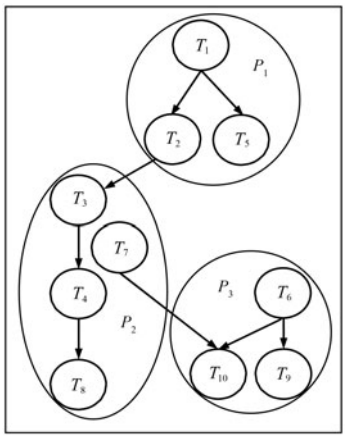


Fig. 7 Final partitioning

## 6 Experiments

In this section, we used four algorithms in order to divide each of below examples. We used the initial network flow algorithm[10, 11], the enhance network flow[12], the list scheduling algorithm[8], and the proposed algorithm. In each case, we evaluated the performance of each algorithm in terms of total communication cost, whole latency of the graph and run time of the algorithm. The host computer was a Core Duo CPU, memory 1 GB running at Windows XP. The graphs shown in Table 2 were chosen to be implemented on FPGA Vertex-II XC2V1000. The Vertex-II XC2V1000 has the following characteristics, as given in Table 3 below.

Table 2 Benchmark characteristics

| DFGs | Nodes | Edges | Area (CLBs) |
|---|---|---|---|
| DCT 4 × 4 | 224 | 256 | 8045 |
| DCT 16 × 16 | 1929 | 2304 | 13919 |
| 16-FFT | 203 | 228 | 5226 |
| 64-FFT | 1287 | 664 | 10098 |

Table 3 Characteristics of the device

| Descriptions | Values |
|---|---|
| Number of lines | 40 |
| Number of Columns | 32 |
| Size (CLB) | 1280 |
| Input/output ports | 432 |
| Configuration time $C_T$ | 7.73 ms |

The discrete cosine transform (DCT) is the most computationally intensive part of the CLD algorithm, that is why it has been chosen to be implemented in hardware. The model proposed by [16] is based on 16 vector products. Thus, the entire DCT is a collection of many nodes, where each node is a vector product. 16-FFT and 64-FFT (fast Fourier transform) are 16 points and 64 points of FFT, respectively, have important roles in analysis, design, and implementation of discrete-time signal processing algorithms and systems. Table 2 gives the characteristics of $4\times4$ DCT, $16 \times 16$ DCT, 16-FFT and 64-FFT task graphs.

Let M.C cost denote maximum communication cost. Table 4 gives different solutions provided by the list scheduling, the initial network flow technique, the enhance network flow and the proposed algorithm. First, our algorithm has always the lowest number of partitions. In fact, as configuration time of currently dynamically reconfigurable hardware is very large. Thus, the configuration overhead will be a problem because the configuration time mainly occupies the time required to switch a partition to another partition. Therefore, since our algorithm has the lowest number of partitions, it has the lowest latency. Results show an average improvement of 20.5 % in terms of design latency. Second, Table 4 shows that our partitioning algorithm minimizes communication overhead between partitions for dynamically reconfigurable hardware. The results show an average improvement of 28.87 %, 13.18 %, and 6.31 % for actual applications, compared with three conventional algorithms. As conclusion, our algorithm has a good tradeoff between computation and communication. Hence, our algorithm can be qualified to be a good temporal partitioning candidate. In fact, an optimal partitioning algorithm needs to balance computation required for each partition and reduce communication required between partitions so that mapped applications can be executed faster on dynamically reconfigurable hardware.

## 7 Conclusions

Today′s large and complex designs are now commonly implemented in FPGAs, however designer suffers principally from the time needed due to communication overhead, which is still relatively high. A high reconfiguration time may lead to impractical design mainly when designer focuses on minimizing the overall communication of the design partition. For that reason, we have developed in this paper a typical temporal partitioning algorithm to reduce the communication between design partitions. In fact, our algorithm uses the eigenvectors of the graph to find the best schedule of nodes that minimizes the communication between design partitions of the graph. In addition, to show the effectiveness of our algorithm, the algorithm is experimented on benchmark circuits such as DCT, FFT task graphs. The studied evaluation cases show that the proposed algorithm provides very significant results terms of communication cost and latency versus other well known algorithms used in the temporal partitioning field.

## Appendix: Proofs of lemmas

**Proof of Lemma 1.**

**Part 1.** By the definition of the matrix $D$

$$X^{\mathrm{T}}LX = X^{\mathrm{T}}DX - X^{\mathrm{T}}DX - X^{\mathrm{T}}WX = \sum_{i=1}^{n} D_i x_i^2 -$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} W_{i,j}x_i x_j = \frac{1}{2}\sum_{i=1}^{n}(D_i + D_i)x_i^2 =$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} W_{i,j}x_i x_j =$$

$$\frac{1}{2}\left(\sum_{i=1}^{n} D_i x_i^2 - 2\sum_{i=1}^{n}\sum_{j=1}^{n} W_{i,j}x_i x_j + \sum_{j=1}^{n} D_j x_j^2\right)$$

$$\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} W_{i,j}(x_i - x_j)^2.$$

Table 4 Design results

| | Proposed algorithm | List scheduling | Initial network flow | Improved network flow | Improvement versus list scheduling | Improvement versus initial network flow | Improvement versus improved network flow |
|---|---|---|---|---|---|---|---|
| Graph | 4 × 4 DCT Task graph | 4 × 4 DCT Task graph | 4 × 4 DCT Task graph | 4 × 4 DCT Task graph | 4 × 4 DCT Task graph | 4 × 4 DCT Task graph | 4 × 4 DCT Task graph |
| Number of partitions | 7 | 9 | 9 | 9 | – | – | – |
| *TCcost* | 570 | 744 | 634 | 589 | 23.38 % | 10.09 % | 3.22 % |
| M.C cost | 110 | 105 | 83 | 81 | – | – | – |
| Whole latency | 5.770 ns + 7× $C_T \cong 7 \times C_T$ | 4770 ns + 9× $C_T \cong 9 \times C_T$ | 4395 ns + 9× $C_T \cong C_T$ | 4570 ns 9× $C_T \cong 9 \times C_T$ | 22 % | 22 % | 22 % |
| Run time | 0.2 s | 0.12 s | 0.12 s | 0.12 s | – | – | – |
| Graph | 16 × 16 DCT Task graph | 16 × 16 DCT Task graph | 16 × 16 DCT Task graph | 16 × 16 DCT Task graph | 16 × 16 DCT Task graph | 16 × 16 DCT Task graph | 16 × 16 DCT Task graph |
| Number of partitions | 11 | 15 | 15 | 15 | – | – | – |
| *TCcost* | 2023 | 3106 | 2378 | 2193 | 34.86 % | 14.92 % | 7.75 % |
| M.C cost | 365 | 297 | 265 | 228 | – | – | – |
| Whole latency | 8420 ns + 11× $C_T \cong 11 \times C_T$ | 6610 ns + 15× $C_T \cong 15 \times C_T$ | 6420 ns + 15× $C_T \cong 15 \times C_T$ | 7730 ns + 15× $C_T \cong 15 \times C_T$ | 26 % | 26 % | 26 % |
| Run time | 2 s | 1.55 s | 1.55 s | 1.55 s | – | – | – |
| Graph | 16-FFT Task graph | 16-FFT Task graph | 16-FFT Task graph | 16-FFT Task graph | 16-FFT Task graph | 16-FFT Task graph | 16-FFT Task graph |
| Number of partitions | 5 | 6 | 6 | 6 | – | – | – |
| *TCcost* | 343 | 488 | 392 | 363 | 29.71 % | 12.5 % | 5.50 % |
| M.C cost | 92 | 93 | 77 | 72 | – | – | – |
| Whole latency | 4730 ns + 5× $C_T \cong 5 \times C_T$ | 3855 ns + 6× $C_T \cong 6 \times C_T$ | 3665 ns + 6× $C_T \cong 6 \times C_T$ | 3255 ns + 6× $C_T \cong 6 \times C_T$ | 4.85 % | 4.85 % | 4.85 % |
| Run time | 0.2 s | 0.09 s | 0.09 s | 0.09 s | – | – | – |
| Graph | 64-FFT Task graph | 64-FFT Task graph | 64-FFT Task graph | 64-FFT Task graph | 64-FFT Task graph | 64-FFT Task graph | 64-FFT Task graph |
| Number of partitions | 9 | 11 | 11 | 11 | – | – | – |
| *TCcost* | 1432 | 1976 | 1689 | 1570 | 27.53 % | 15.21 % | 8.78 % |
| M.C cost | 189 | 229 | 185 | 172 | – | – | – |
| Whole latency | 7250 ns + 9× $C_T \cong 9 \times C_T$ | 6760 ns + 11× $C_T \cong 11 \times C_T$ | 6520 ns + 11× $C_T \cong 11 \times C_T$ | 6550 ns + 11× $C_T \cong 11 \times C_T$ | 18 % | 18 % | 18 % |
| Run time | 2 s | 1.31 s | 1.31 s | 1.31 s | – | – | – |
| Average improvement in communication cost | | | | | 28.87 % | 13,18 % | 6.31 % |
| Average improvement in latency | | | | | 20.5 % | 20.5 % | 20.5 % |

**Part 2.** The symmetry of $L$ follows directly from the symmetry of $W$ and $D$. The positive semi-definiteness is a direct consequence of Part 1, which shows that $X^{\mathrm{T}}LX \geqslant 0$ for all $X = \{x_1, x_2, \cdots, x_n\} \in \mathbf{R}^n$.

**Part 3.** This is a direct consequence of Part 1 and Part 2. □

**Proof of Lemma 2.** According to the properties of Laplacian matrix

$$X_m^{\mathrm{T}}LX_m = X_m^{\mathrm{T}}DX_m = X_m^{\mathrm{T}}WX_m = \sum_{i=1}^{n} D_i x_m^2(i) \sum_{i=1}^{n}\sum_{j=1}^{n} W_{i,j}x_m(i)x_m(j) =$$

$$\frac{1}{2}\sum_{i=1}^{n}(D_i + D_i)x_m - \sum_{i=1}^{n}\sum_{j=1}^{n} W_{i,j}x_m(i)x_m(j) =$$

$$\frac{1}{2}\left(\sum_{i=1}^{n}(D_i + D_i)x_m^2(i) - 2\sum_{i=1}^{n}\sum_{j=1}^{n} W_{i,j}x_m(i)x_m(j) + \sum_{j=1}^{n} D_j x_m^2(j)\right) =$$

$$\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} W_{i,j}(x_m(i) - x_m(j))^2 =$$

$$\frac{1}{2}\left(\frac{\sum_{T_i \in P_m; T_j \in |\overline{P_m}|} W_{i,j}}{|P_m|} + \frac{\sum_{T_i \in P_m; T_j \in |\overline{P_m}|} W_{i,j}}{|\overline{P_m}|}\right) =$$

$$\frac{|V|}{2}\left(\sum_{i=1}^{k}\frac{\sum_{T_i\in P_m;T_j\in|\overline{P_m}|}W_{i,j}}{|P_m||\overline{P_m}|}\right)\Rightarrow\sum_{m=1}^{k}X_m^tLX_m=$$
$$\frac{V}{2}\left(\sum_{m=1}^{k}\frac{\sum_{T_i\in P_m;T_j\in|\overline{P_m}|}W_{i,j}}{|P_m||\overline{P_m}|}\right)=TCCost.$$

□

**Proof of Lemma 3.** The $ij$-th element of the matrix $X_pX_p^{\mathrm{T}}$ is $\sum_{m=1}^{k}X_m(i)X_m(f)$. The term $X_m(i)X_m(j)$ will be non-zero if and only if both $T_i$ and $T_j$, are in $P_m$, hence the sum is $1/|P_m|$ when $T_i$ and $T_j$ are in the same partition; 0 otherwise. □

## References


[1] C. Bobda. *Introduction to Reconfigurable Computing: Architectures, Algorithms and Applications*, Germany: Springer Publishers, 2007.

[2] J. M. P. Cardoso. On combining temporal partitioning and sharing of functional units in compilation for reconfigurable architectures. *IEEE Transactions on Computers*, vol. 52, no. 10, pp. 1362–1375, 2003.

[3] C. Tanougast, Y. Berviller, P. Brunet, S. Weber, H. Rabah. Temporal partitioning methodology optimizing FPGA resources for dynamically reconfigurable embedded real-time system. *Microprocessors and Microsystems*, vol. 27, no. 3, pp. 115–130, 2003.

[4] A. D. Cabrol, T. Garcia, P. Bonnin, M. Chetto. A concept of dynamically reconfigurable real-time vision system for autonomous mobile robotics. *International Journal of Automation and Computing*, vol. 5, no. 2, pp. 174–184, 2008.

[5] W. Y. Wu, Z. X. Zhao. Realization of reconfigurable virtual environments for virtual testing. *International Journal of Automation and Computing*, vol. 2, no. 1, pp. 25–36, 2005.

[6] B. Ouni, R. Ayadi, M. Abid. Novel temporal partitioning algorithm for run time reconfigured systems. *Journal of Engineering and Applied Sciences*, vol. 3, no. 10, pp. 766–773, 2008.

[7] B. Ouni, A. Mtibaa, E. B. Bourennane. Scheduling approach for run time reconfigured systems. *International Journal of Computer Sciences and Engineering Systems*, vol. 3, no. 4, pp. 335–340, 2009.

[8] S. Trimberger. Scheduling designs into a time-multiplexed FPGA. In *Proceedings of the 1998 ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, ACM, Monterey, USA, pp. 153–160, 1998.

[9] H. Q. Liu, D. F. Wong. A graph theoretic algorithm for schedule compression in time-multiplexed FPGA partitioning. In *Proceedings of IEEE/ACM International Conference on Computer-aided Design*, IEEE, San Jose, USA, pp. 400–405, 1999.

[10] H. Q. Liu, D. F. Wong. Network flow based circuit partitioning for time-multiplexed FPGAs. In *Proceedings of IEEE/ACM International Conference on Computer-aided Design*, IEEE, pp. 497–504, 1998.

[11] H. Q. Liu, D. F. Wong. Network flow based multi-way partitioning with area and pin constraints. *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems*, vol. 17, no. 1, pp. 50–59, 1998.

[12] W. K. Mak, E. F. Y. Young. Temporal logic replication for dynamically reconfigurable FPGA partitioning. *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems*, vol. 22, no. 7, pp. 952–959, 2003.

[13] Y. C. Jiang, J. F. Wang. Temporal partitioning data flow graphs for dynamically reconfigurable computing. *IEEE Transactions on Very Large Scale Integration Systems*, vol. 15, no. 12, pp. 1351–1361 ,2007.

[14] J. Shi, J. Malik. Normalized cuts and image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[15] P. Biswal, J. R. Lee, S. Rao. Eigenvalue bounds spectral partitioning and metrical deformations via flows. In *Proceedings of the 49th IEEE Symposium on Foundations of Computer Science*, IEEE, Philadelphia, USA, pp. 751–760, 2008.

[16] M. Kaul, R. Vemuri. Design-space exploration for block-processing based temporal partitioning of run-time reconfigurable systems. *Journal of VLSI Signal Processing Systems for Signal Image and Video Technology*, vol. 24, no. 2, pp. 181–209, 2000.



**Ramzi Ayadi** received his M. Sc. degree in electronics and microelectronics from the University of Monastir, Tunisia in 2007. He is currently completing his Ph. D. degree at the Electronic and Micro-Electronic Laboratory, Faculty of Sciences of Monastir.

His research interests include high level synthesis, methodologies development for reconfigurables architectures.

E-mail: ramzi_ayadi@yahoo.fr (Corresponding author)



**Bouraoui Ouni** received his licence and his master respectively in 1999 and 2001. He completed his thesis in 2008 from the Faculty of Science of Monastir, Tunisia. Since 2002, he has been an assistant professor in digital electronic at the High Institute of Informatics and Telecommunication of Hamman Sousse and the National School of Engineering of Sousse.

His research interests include high level synthesis and methodologies development for reconfigurable architectures.

E-mail: bouraoui.ouni@fsm.rnu.tn



**Abdellatif Mtibaa** is currently professor in micro-electronics and hardware design with Electrical Department at the National School of Engineering of Monastir, Tunisia and head of Circuits Systems Reconfigurable-ENIM-Group at Electronic and Microelectronic Laboratory. He holds a diploma in electrical engineering in 1985 and received his Ph. D. degree in electrical engineering in 2000. He has authored/co-authored over 100 papers in international journals and conferences. He served on the technical program committees for several international conferences. He also served as a co-organizer of several international conferences.

His research interests include system on programmable chip, high level synthesis, rapid prototyping and reconfigurable architecture for real-time multimedia applications.

E-mail: abdellatif.mtibaa@enim.rnu.tn