International Journal on New Computer Architectures and Their Applications (IJNCAA) 1(1): 112-122
The Society of Digital Information and Wireless Communications, 2011 (ISSN 2220-9085)

An efficient temporal partitioning algorithm to minimize Communication Cost for reconfigurable computing Systems

Ramzi. Ayadi[1], Bouaoui. Ouni[1], Abdellatif. Mtibaa[1]
[1]Laboratory of Electronic and Microelectronic, Faculty of science at Monastir, Monastir 5000, Tunisia
{ramzi_ayadi, bouraoui.ouni, abdellatif.mtibaa} @yahoo.fr, @fsm.rnu.tn, @enim.rnu.tn

ABSTRACT

In reconfigurable computing systems, full reconfigurable FPGA are evolving rapidly, due to their flexibility and high performance. In this paper, we focus on communication cost between partitions in order to develop an algorithm to solve temporal partitioning problems for full reconfigurable architecture. In fact, this algorithm optimizes the transfer of data required between design partitions. The proposed algorithm was tested on several examples on the Xilinx Virtex-II pro. The results show significant reduction in the communication cost compared with others famous approaches used in this field

KEYWORDS

Temporal partitioning, data flow graph, full reconfigurable architectures, FPGA

1 INTRODUCTION

The temporal partitioning problem [1][2][3] can be formulated as a graph-based problem. A program or application can be represented by a data flow graph (DFG). A DFG is a directed acyclic graph G= (V, E), where V is the set of nodes $|V|$ = n = number of nodes in G and E is a set of edges. Each node $T_i \in V$ represents a functional operation, correspondingly $A(T_i)$, represents the operation size. A directed edge $e_{i,j} \in E$ exists if there is data dependency between node T_i and T_j. We define the weight $\alpha_{i,j}$ of $e_{i,j}$ as the amount of data transferred from T_i toT_j. A temporal partitioning *P* of the graph G = (V, E), is its division into some disjoints partitions such as: $P = \{P_1. \ . .P_k\}$. The temporal partitioning problem has been formulated as follows.

- Inputs: given a data flow graph G = (V, E)
- Constraints:

1) $V = \bigcup_{i=1}^{k} T_i$. Where K is a number of partitions
2) All dependency constraint relations are satisfied for all K partitions, let $Dep(T_i)$ denote the dependency constraint of a node T_i. For tow nodes T_i and T_j , we define $Dep(T_i) \leq Dep(T_j)$ if T_i must be scheduled no later than T_j.
3) $A(P_i) \leq A(H), 1 \leq i \leq k.$ Where $A(P_i)$ denoted the area of partition P_i and $A(H)$ denoted the total area of the reconfigurable processing unit (RPU).
4) $\forall P_k \in P, e_{ij} \in E$, we have

$$\frac{1}{2}\left(\sum_{(e_{ij} \cap P_k \neq \emptyset)\ and\ (e_{ij} - P_k \neq \emptyset)} \alpha_{ij}\right) \leq T(H)$$

Where $T(H)$ number of programmable input/outputs (I/Os) per device. We extend the ordering relation $\leq$ to *P* as

follow: $P_i \leq P_j \Leftrightarrow e_{ij} \in E$ with $T_i \in P_i$ and $T_j \in P_j$ either $T_i \leq T_j$ or $\leq$ is not defined for T_i and T_j. The partition P is ordered $\Leftrightarrow$ an ordering relation $\leq$ exists for P. An ordered partitioning is characterized by the fact that for a pair of partitions, one should be always implemented after the other with respect to any scheduling relation.

- Objectives:

Several objective functions can be defined for the temporal partitioning problem. One objective could be the minimization of the number of partitions to reduce the overall reconfiguration overhead. Another objective could be the minimization of the computation time. This can be expressed for example through the minimization of the maximum computational delay across all the partitions. A third objective could be the communication cost of the design. This aim can be reached by minimizing the transfer of data required between design partitions.

The approaches described above have used Full Reconfigurable architectures (FRA) shown in figure 1, as target architecture. FRA has been constructed from one or more general purpose processors (GPP), an external memory and a reconfigurable processing unit (RPU). In fact, designers have used the temporal partitioning approach to divide the application into temporal partitions, which are configured one after the one on the target RPU. The first partition receives input data, performs computations and stores the intermediate data into an on-board memory. The device is then reconfigured for the next partition, which computes results based on intermediate data from the previous partition. A controller interacts with both the reconfigurable hardware and the memory and is used to load new configuration.

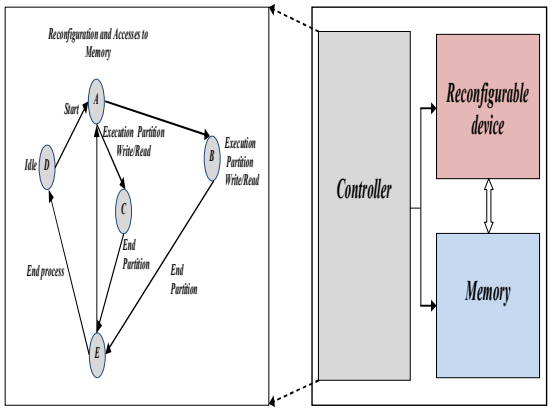

Figure1: Full reconfigurable architecture

In this paper, we focus on communication cost between partitions in order to develop an algorithm to solve temporal partitioning problem for full reconfigurable architecture. In fact, this algorithm optimizes the transfer of data required between design partitions and the reconfiguration overhead.

2 RELATED WORKS

In the literature, many methods have developed by different authors to solve the temporal partitioning problem. In [4][5][6] authors used traditional scheduling methods, such as list scheduling. The idea behind the list scheduling approach is first to place all the nodes of a graph representing the problem to be solved in a list. A new partition (also called configuration) is built stepwise by removing nodes from the list and allocating them to the partition until the size of the partition reached a given size limit (the size of the FPGA). A new partition is then created and the process is repeated until all the nodes from the list are placed in partitions. Others authors extended existing scheduling of high-level

synthesis [7][8]. In [9][10] the authors used ILP algorithm. The ILP is a mathematical method for determining a way to achieve the best outcome, such as lowest latency. The main problem of the ILP approach is its high execution time; therefore, the algorithm can only be applied to small examples. In [11] authors combined the force directed scheduling (FDS) algorithm and network flow algorithm to reduce the whole latency and the communication cost at the same time. In [12] author used leveling node method to determine the communication cost. However, for each end of stage the method is not a min cut just only a leveling cut. Also, the network flow algorithm has been used to reduce the communication cost across temporal partitions in [12][13]. The first network flow algorithms has been used in [12][13][14] and improved in [3]. 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 following description shows the initial network algorithm as presented in [12][13].

Begin
1. Construct graph G' from graph G by net modeling
2. Pick a pair of node s and t in G' as source and sink
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
4. If (lr ≤ w(X) ≤ ur) then stops and return C as solution
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
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

Where: w(X) is the total area of all nodes in X; Lr = (1-ε) Rmax , Rmax is the area of the device; ur =(1-ε) Rmax; ε=0,05, S is the source node, t is the sink node. Let us consider the graph G of figure 2.a, after following the net modeling steps, such as presented in [12][13], the new graph G' of figure 2.b is obtained

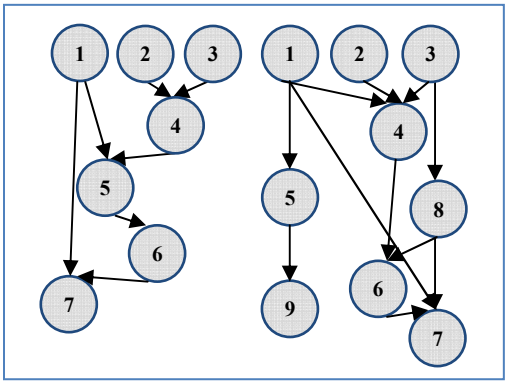

Figure 2.a: Graph G Figure 2.b: graph G'

Let us assume 200 CLBs be the area of the device, 100 CLBs be the area of the multiplier, 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 figure 2.a. the result is shown in figure 3, the network flow algorithm puts nodes T_2, T_3, T_4 in partition P_1, nodes T_1, T_5, T_6 in partition P_2 and node T_7 in partition P_3.

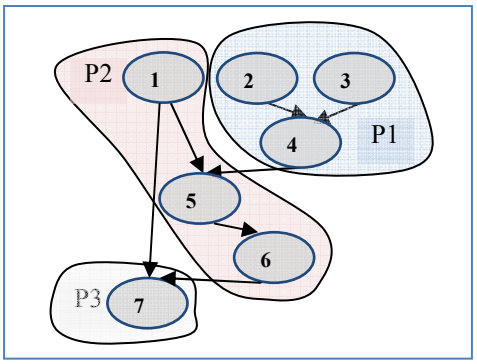

Figure 3: Temporal partitioning

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 become difficult to handle. Further, the network flow algorithm, is a heuristic algorithm, in fact there is no a mathematical model behind him.

3 DEFINITIONS

3.1 Definition 1

Given a data flow graph G = (E, V), we define:

- The (n x n) weighted adjacency matrix W(G) as follows; n is the number of nodes in G:

$$W_{ij} = e_{ij}$$
$$W_{ii} = 0$$

- The (n x n) degree matrix D(G) as follows:

$$D_{ii} = \deg(T_i) = \sum_{i=1}^{n} W_{ij}$$
$$D_{ij} = 0$$

- The (n x n) Laplacian matrix L(G) as follows:

$$L(G) = D(G) - W(G)$$

3.2 Definition 2

An n-vector ϑ is an eigenvector of L(G) with eigenvalue λ if and only $L(G)\lambda = \vartheta\lambda$. We denote the set of eigenvectors of L (G) by $\vartheta_1, \vartheta_2, \ldots, \vartheta_n$ with corresponding eigenvalues $\lambda_1 \leq \lambda_n \leq \mu \ldots \ldots \leq \lambda_n$ .The n x n eigenvector matrix $U_n = (\vartheta_{ij})$ has columns $\vartheta_1, \vartheta_2, \ldots, \vartheta_n$and the n x n eigenvalue matrix $V_n = (\lambda_{ii})$ has diagonal entries, and 0 entries elsewhere.

We assume that the eigenvectors are normalized, i.e. for $1 \leq i \leq n$, $\vartheta_i \vartheta_i^T = I$. The eigenvectors of L(G) have many desirable properties, including [15]:

- The eigenvectors are all mutually orthogonal; hence they form a basis in n-dimensional space.
- $\lambda_1 = 0 \; and \; \vartheta_1 = \left[\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}, \ldots \ldots, \frac{1}{\sqrt{n}}\right]^T$
- L(G) has n non-negative, real-valued eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq, \ldots, \leq \lambda_n$
- $L(G)\vartheta_i = \lambda_j \vartheta_j$
- For every vector $X = \{x_1, x_2, \ldots \ldots, x_n\} \in R^n$; we have :

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

3.3 Definition 3

Given a temporal partitioning of *G = (E, V)* into k disjoint partitions *P = {P1, P2...Pk}*; the communication cost, *CCost (Pm)*, of partition P_m has been defined in [13] as follow:

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

This implies that:

$$TCCost = \sum_{i=1}^{K} CCost(P_m)$$

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

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

4 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 in optimal number of temporal partitions that minimize the communication cost between partitions of the graph while respecting all constraints.

Our algorithm is composed by two main steps. The first step aims to find an initial partitioning P_{in} that minimizes communication cost of the graph. Next, if the area constraint is satisfied after the first step then we adopts 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 constrain. If the second step cannot find a feasible scheduling then we relax the number of partition by one and the algorithm goes to the first step. And, we restart to find a feasible solution in the new number partitions.

4.1 First step: initial partitioning

As shown in Eq. (2) the total communication cost minimization problem can be solved by minimizing the total cut size between design partitions. In this section, we present a good solution for this problem.

Lemma1:

Given a temporal partitioning of *G = (E, V)* into k disjoint partitions *P = {P1, P2...Pk}* .*We* define the indicator vector X_m as follow:

$X_m(i) = \{X_m(1), X_m(2), \ldots., X_m(n)\}^t$, where m = 1, 2,..., k and i = 1,2,…n are defined as:

$X_m(i) = \frac{1}{\sqrt{|\overline{P_m}|}}$ if $T_i \in P_m$; 0 otherwise.

We have: $TCCost = \sum_{i=1}^{K} X_m^t \, L X_m$ (3)

Poof:

According Properties of Laplacian matrix

$$X_m^t L X_m = X_m^t D X_m = X_m^t W X_m$$

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

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

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

$$= \frac{1}{2} \left(\sum_{i=1}^{n} (D_i + D_i) x_m^2(i) \right.$$

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

$$\left. + \sum_{j=1}^{n} D_j \, x_m^2(j) \right)$$

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

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

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

$$\Longrightarrow \sum_{m=1}^{k} X_m^t L X_m$$

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

$$= TCCost$$

Observation 1:
Using equation 3, the problem of communication cost minimization can be expressed as:

Minimize(TCCost)=>Minimize(trace ($X_m^t L X_m$)

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(G) as columns

Lemma2:
Given a (n×n) matrix ***Mp*** as follow:

$M_{i,j} = \frac{1}{|P_m|}$ if T_i and $T_j \in P_m$; 0 otherwise

We have: $X_m^t L X_m = M_p$

Proof:

The ij^{th} of $X_p X_p^t$ is $\sum_{m=1}^{k} X_m(i)\, X_m(j)$. The term $X_m(i)X_m(j)$ will be non-zero if and only if both T_i and T_j, are in $\boldsymbol{P_m}$, hence the sum is $\frac{1}{|P_m|}$ when T_i and T_j are in the same partition; 0 otherwise.

Lemma3:
To calculate the lower number of partitions required to obtain solution we divide the area of all nodes, by the available reconfigurable resource. In other words, given a graph G = (V, E) partitioned into K disjoints temporal partitions; $P = \{P_1, P_2 \ldots P_k\}$; the lower number of temporal partitions K_{min} is: $\frac{A(G)}{A(H)}$, where A(G) is the area of the graph and A(H) is the area of the device.

Proof:
Given a function F(K) defined as follows:

$$F(k) = A(G) - \sum_{i=1}^{K} A(P_i)$$

Then, Min(F(k)) correspond to Max($\sum_{i=1}^{K} A(P_i)$).

Or $\forall\ P_i \in P$;

$$\sum_{i=1}^{K} A(P_i) \leq \sum_{i=1}^{K} MaxA(P_i) \Rightarrow$$

$$Max\left(\sum_{i=1}^{K} A(P_i)\right) = \sum_{i=1}^{K} MaxA(P_i) = KMax\big(A(P_i)\big) \Rightarrow$$

$$K_{min} = \frac{A(G)}{MaxA(P_i)}$$

Or $\forall\ P_i \in P$;

$$A(P_i) \leq A(H) \Rightarrow MaxA(P_i) = A(H)$$

$$\Rightarrow K_{min} = \frac{A(G)}{A(H)} \quad (4)$$

The above descriptions are summarized by the following steps, form step 1 to step 7

1) Compute the minimum number of partitions K= K_{min} = [*Area*(G) / *Area*(*H*)]

2) Compute the laplacian matrix L(G) of G

3) Compute k lowest eigenvalues of L(G)

4) Construct the (n x k) matrix X_P that have the K eigenvectors as columns.

5) Compute $Z = X_P X^t_P$

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

7) Generate the initial partitioning from matrix M_p
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)

4.2 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 Pi to P_j or inversely until the satisfaction of the area constraint. The balance of nodes is based on the force $F(T_i,P_i \rightarrow P_j)$ associated with partition P_i on a node T_i to be scheduled into partition P_j and on the force $F(T_i,P_j \rightarrow P_i)$ 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:

$$F(T_i,P_i \rightarrow P_j) = \delta_1(T_i) * OF(T_i) \quad (11)$$

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

$$OF(T_i) = (Nu(T_i) + 1) \quad (12)$$

Given tow nodes $T_i \; and \; T_j \in P_i$

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

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

$$F(T_i,P_j \rightarrow P_i) = \delta_2(T_i) * InF(T_i) \quad (14)$$

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

$$InF(T_i) = (Nq(T_i) + 1) \quad (15)$$

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

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

Where: $\emptyset_{i,j} = 1$ 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 EXPRIMENTS:

In our experiences, we used four approaches, list scheduling [6], initial network flow [13], improved network flow [3] and the proposed algorithm. In our experiences, we evaluated the performance of each algorithm in term of total communication cost, whole latency of the graph and run time of the algorithm. The graphs shown in table 2 and table 4 were chosen to be implemented on FPGA Vertex-II XC2V1000. The Vertex-II XC2V1000 has the following characteristics, table 1:

Table 1: characteristics of the device

Number of lines	**40**
Number of Columns	**32**
Size (CLB)	**1280**
input / output ports	**432**
Configuration time C_T	**7,73 ms**

We have conducted two kinds of experiment. In the first experiment we considered the 4x4 and 16x16 DCT data flow graph. In the second one, we considered the combinational circuits c3540 and c6288:

The figure 4 shows the Color Layout Descriptor ''CLD'' is a low-level visual descriptor that can be extracted from images or video frames. The process of the CLD extraction consists of four stages: Image partitioning, selection of a single representative color for each block, DCT transformation and non linear quantization and Zig-Zag scanning.

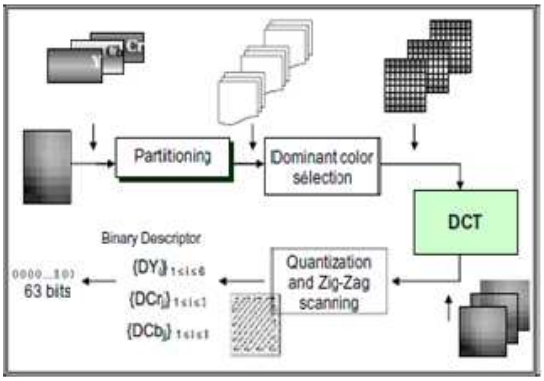

Figure 4: Block diagram of the CLD extraction

Since DCT is the most computationally intensive part of the CLD algorithm, it has been chosen to be implemented in hardware, and the rest of subtasks (partitioning, color selection, quantization, zig-zag scanning and Huffman encoding) were chosen for software implementation. The model proposed by [16] is based on 16 vector products. Thus, the entire DCT is a collection of 16 tasks, where each task is a vector product as presented in Figure 5.

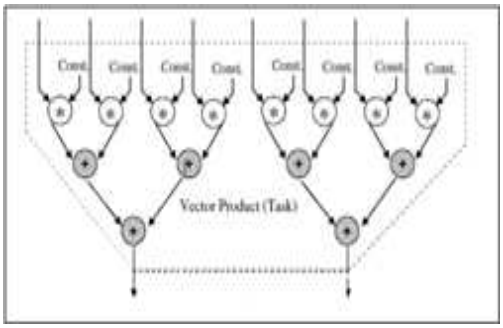

Figure 5: vector products

There are two kinds of tasks in the task graph.'' T1'' and ''T2'', whose structure is similar to vector product, but whose bit widths differ. Table 2 gives the characteristic of 4x4 DCT, 16x16 DCT task graphs.

Table 2: Benchmark characteristics (DCT task graphs).

DFGs	Nodes	Edges	Area (CLBs)
DCT 4X4	224	256	8045
DCT 16X16	1929	2304	13919

The c3540 and c6288 are benchmarks of the combinational circuit commonly used to test CAD algorithms. In these circuits, every gate corresponds to a node which has an area size as shown in Table 3. We can easily transform a combinational circuit into a DFG with weighted nodes by a transform program. Table 4 gives the characteristic of c3540 and c6288 task graphs.

Table 3: Area of the logical gates expressed in circuit.

Logical gate	Area (CLBs)
Buffer	2
INT	3
AND	5
OR	7
NAND	8
NOR	12
XOR	14
XNOR	18

Table4: Benchmark characteristics (combinational circuit).

DFGs	Nodes	Edges	Area (CLBs)
c3540	1038	1016	6426
c6288	2247	2140	10644

The table 5 and table 6 gives the different solutions provided by the list scheduling, the initial network flow technique, the enhance network flow and the proposed algorithm. Firstly, our algorithm has always the lowest number of partitions. In fact, as configuration time of currently dynamically reconfigurable hardware is too 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 tem of design latency for the DCT task graph and 14, 16% for the combinational circuit task graph. Secondly, the table 5 and table 6 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 for the DCT task graph and 27,89% , 6,145% and 1, 51% for the combinational circuit task graph.

As conclusion our algorithm has a good trade-off 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.

6 CONCLUSION:

Dynamically reconfigurable computing systems have the potential for achieving high performance at a relatively low cost for a wide range of applications. In this paper, we have proposed a new temporal partitioning algorithm for reconfigurable computing systems to reduce maximum communication cost. Our algorithm is composed by two main steps. The first step aims to minimize the transfer of data required between design partitions. To satisfy area constraints, we use the balance of nodes technique. The experiments on benchmark circuits such as DCT combinational circuits task graphs have shown the effectiveness of the proposed algorithm.

Table 5: Design results (DCT task graphs).

Graph	4X4 DCT Task graph						
Algorithms	*Proposed algorithm*	*List scheduling*	*Initial Network flow*	*improved Network flow*	*Improvement Versus List scheduling*	*Improvement Versus Initial Network flow*	*Improvement Versus improved Network flow*
Number of Partitions	7	9	9	9			
T.C cost	570	744	634	589	23,38%	10,09%	3,22%
M.C cost	110	105	83	81			
Whole latency	5,770 ns + 7* $C_T \cong 7* C_T$	4770 ns+ 9* $C_T \cong 9* C_T$	4395ns + 9* $C_T \cong 9* C_T$	4570 ns 9* $C_T \cong 9* C_T$	22%	22%	22%
Run time	0,2 sec	0,12 sec	0,12 sec	0,12 sec			
Graph	16X16 DCT Task graph						
Number of Partitions	11	15	15	15			
T.C cost	2023	3106	2378	2193	34,86%	14,92%	7,75%
M.C cost	365	297	265	228			
Whole latency	8420 ns + 11* $C_T \cong 11* C_T$	6610 ns+ 15* $C_T \cong 15* C_T$	6420ns+15* $C_T \cong 15* C_T$	7730ns+15* $C_T \cong 15* C_T$	26%	26%	26%
Run time	2 sec	1,55 sec	1,55 sec	1,55 sec			
Average improvement in communication cost					28,87%	13, 18%	6, 31%
Average improvement in latency					20,5%	20,5%	20,5%

Table 6: Design results (combinational circuit task graph).

Graph	c3540 Task graph						
Algorithms	*Proposed algorithm*	*List scheduling*	*Initial Network flow*	*improved Network flow*	*Improvement Versus List scheduling*	*Improvement Versus Initial Network flow*	*Improvement Versus improved Network flow*
Number of Partitions	4	5	5	5			
T.C cost	550	783	588	561	29,75%	6,46%	1,96%
M.C cost	147	171	128	126			
Whole latency	3575 ns +4* $C_T \cong 4* C_T$	4240 ns+ 5* $C_T \cong 5* C_T$	4240ns + 5* $C_T \cong 5* C_T$	4240 ns 5* $C_T \cong 5* C_T$	20%	20%	20%
Run time	0,2 sec	0,82 sec	0,82 sec	0,82 sec			
Graph	c6288 Task graph						
Number of Partitions	11	12	12	12			
T.C cost	829	1121	886	835	26,04%	6,43%	1,07%
M.C cost	80	132	89	86			
Whole latency	6860 ns + 11* $C_T \cong 11* C_T$	6780 ns+ 12* $C_T \cong 12* C_T$	6780ns+12* $C_T \cong 12* C_T$	6780ns+12* $C_T \cong 12* C_T$	8,33%	8,33%	8,33%
Run time	2 sec	1,97 sec	1,97 sec	1,97 sec			
Average improvement in communication cost					27,89%	6, 145%	1, 51%
Average improvement in latency					14,16%	14,16%	14,16%

7 REFERENCES

1. Bobda,C "Introduction to Reconfigurable Computing Architectures, Algorithms, and Applications", Book, Springer Publishers (2007).
2. Joao M.P. Cardoso, "On Combining Temporal Partitioning and Sharing of Functional Units in Compilation for Reconfigurable Architectures", IEEE Trans. Computers, Vol. 52 No. 10, 2003, pp : 1362-1375
3. Yung-Chuan J and Yen-Tai L "Temporal partitioning data flow graphs for dynamically reconfigurable computing" IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Volume 15 , Issue 12 (December 2007), pp: 1351 – 1361
4. Bouraoui Ouni, Ramzi Ayadi, Mohamed Abid, Novel temporal partitioning algorithm for run time reconfigured systems, Journal of Engineering and Applied Sciences (3) (2008).
5. Cardoso JMP. On combining temporal partitioning and sharing of functional units in compilation for reconfigurable architectures. IEEE Trans Comput 2003;52(10):1362–75.
6. S. Trimberger, "Scheduling designs into a time-multiplexed FPGA," in Proc. ACM Int. Symp. Field Program. Gate Arrays, 1998, pp: 153–160.
7. Spillane J, Owen H. Temporal partitioning for partially-reconfigurable field programmable gate. Reconfigurable Architectures Workshop in PS/SPDP'98.
8. Ouni B, Ayadi R, Mtibaa A. Partitioning and scheduling technique for run time reconfigured systems. Int J Comput Aided Eng Technol 2011;3(1):77–91.
9. Jeong Byungil. Hardware software partitioning for reconfigurable architectures. MS theses. School of Elec. Eng. Seoul National University; 1999
10. Wu GM, Lin JM, Chang YW. Generic ILP-based approaches for time multiplexed FPGA partitioning. IEEE Trans Comput – Aided Des 2001;20(10):1266–74.
11. Bouraoui Ouni, Abdellatif Mtibaa, El-Bay Bourennane scheduling approach for run time reconfigured systems, International Journal of Computer Sciences and Engineering Systems (4) (2009).
12. Liu. H and Wong. D. F, "Network flow based circuit partitioning for time-multiplexed FPGAs," in Proc. IEEE/ACM Int. Conf. Comput.- Aided Des., 1998, pp:497–504.
13. Liu. H and Wong. D. F ""Network flow based multi-way partitioning with area and pin constraints" IEEE trans on computer aided design of integrated circuits and systems. Vol 17, NO 1, January 1998, pp: : 50 - 59
14. Wai-Kei M and Young E.F. "Temporal logic replication for dynamically reconfigurable FPGA partitioning," IEEE Trans. Computer-Aided Design, vol. 22, Issue 7, July 2003, pp:952-959.
15. B. Mohar, The Laplacian spectrum of graphs, in: Proceedings of the 6th Quadrennial International Conference on the Theory and Applications of Graphs, 1988, pp. 871-898.
16. Abdellatif Mtibaa, Bouraoui Ouni, Mohamed Abid. An efficient list scheduling algorithm for time placement problem. Comput Electr Eng 2007;33(4):285–98.