A Communication-aware Scheduling Algorithm for Hardware Task Scheduling Model on FPGA-based Reconfigurable Systems

Yingying Sheng
College of Information Science and Engineering, Hunan University, Changsha, China
yysheng@hnu.edu.cn

Yan Liu, Renfa Li, Xiongren Xiao
College of Information Science and Engineering, Hunan University, Changsha, China
{liuyan, lirenfa, xxr}@hnu.edu.cn

***Abstract*—Task scheduling is an important aspect of high performance reconfigurable computing. Most of the heuristics for this NP-hard problem are based on a simple abstract model of FPGA and have little investigation into optimizing data communication which influences the system performance importantly. To solve this problem, a Communication-aware Maximum Adjacent Edges (CA-MAE) algorithm based on new 2D reconfigurable model is proposed, which could reduce communication distance during scheduling and enhance the system performance. The experimental results show that CA-MAE reduces communication cost by 17%.**

***Index Terms*—reconfigurable computing, scheduling model, scheduling algorithm, communication time**

I. Introduction

Partially reconfigurable FPGAs are capable of modifying their logic and interconnect configuration at runtime to execute tasks concurrently. The task will be removed from the FPGA when it finishes executing and then the area it occupied can be reclaimed and reused. In such systems, ROS (Reconfigurable Operating System) needs to provide appropriate management of reconfigurable resources and fast configuration decision [1].

Many algorithms have been proposed to achieve efficient resource management and task scheduling. However, most of them are based on very simple models, which can not reflect real FPGAs accurately [2]. Most of the research efforts were based on the assumption that the hardware tasks are rectangles and can be placed in arbitrary area on the 2D area model of FPGA. Broadly speaking, the programmable circuits of FPGAs consist of three parts: CLB (Configuration Logic Block), IOB (I/O Block) and IR (Interconnect Resource). When a configuration is loaded, the connecting and disconnecting of IR are changed. IR is a very important part in FPGA and mainly reflects the connecting requirement, which cannot be neglected in real model [3]. However, there is little investigation into data communications.

This paper investigates task data dependency and the inter-task data communication. A more realistic model taking the communication resources into consideration is developed, and based on this model a scheduling algorithm considering data communication is proposed to elaborate the way in which these connections influence the communication.

The remainder of this paper is organized as follows: Section 2 cites some related work. Section 3 gives an analysis of scheduling model and communication requirement. In Section 4, CA-MAE algorithm based on the new model is proposed. Section 5 investigates experiment results and conclusion is made in Section 6.

II. Related Work

Bazargan et al. [4] did a pioneering work, in which they dealt with the area allocation problem based on a strip packing model. The authors managed the free area by maintaining a list of free rectangles and allocated a task in a fit empty rectangle which was found out from the list.

Steiger et al. [5] proposed two non-preemptive scheduling algorithms: the Stuffing algorithm and Horizon algorithm, both for 1D and for 2D area models. The Stuffing is an important reference algorithm in the related researches. To deal with the drawbacks of the original 1D Stuffing, many researchers studied it and proposed improvements, for example the 1D Intelligent Stuffing [6].

Iturbe et al. [7] focused on 2D model and presented a solution which defines an infrastructure for executing RC applications under real-time constraints coordinately and

Manuscript received January 3, 2014; revised April 18, 2014; accepted April 25, 2014.

Project number: National High-Tech Research and Development Plan of China under Grant No.2012AA01A301-01, National Natural cience Foundation of China under Grant No.61173036 and No.61300037, and Hunan Provincial Natural Science Foundation of China under Grant No. 12JJ4057.

Corresponding author: Yingying Sheng(yysheng@hnu.edu.cn).

doi:10.4304/jcp.9.11.2552-2558

reliably. Their FAEDF (Finishing Aware Earliest Deadline First) algorithm improves non-preemptive EDF (Earliest Deadline First) by considering the plastic time period between finish time and deadline, as the Stuffing does, i.e. it delays the execution of tasks which could not be allocated in the first instance until enough adjacent free areas were released on the FPGA.

Tabero et al. [8] tried to minimize the area fragmentation by a Look-ahead 3D heuristic which locates tasks next to the borders of the free area. The authors achieved reconfigurable resource management by keeping track of available free areas with a vertex-list structure. 3D-adjacency heuristic is similar to the 3D bin packing problem. In 2009, the authors [9] extended the fragmentation metric and presented a resource management for the 3D FPGA model. In [10], Thomas Marconi et al. proposed a 3DTCS (3D total contiguous surface) heuristic which is also a 3D-adjacency heuristic and places tasks at locations contacting its prior tasks and bold lines as much as possible.

III. Scheduling Model

In reconfigurable systems, FPGA is generally abstracted into two-dimensional rectangle Reconfigurable Logic Device (RLD) with W columns ×H rows RCUs (Reconfigurable Computing Unit) to support spatial resource either in 2D or 3D model. The application to be scheduled is usually represented by a DAG (Directed Acyclic Graph) [11], [12], [13].

A. Task Model

After Hardware/Software partition, a hardware task *T (w, h, a, e, d)* is defined as computing functionality that can be synthesized to digital circuit running on RLD [14]. Parameter *w* and *h* respectively represents the number of RCUs it will occupy in *x* and *y* direction. Parameter *a* represents arrival time, *e* is execution time and *d* is deadline of *T*.

When task *T(a, e, d, w, h)* arrives at time *a*, scheduler searches RLD to find an appropriate allocation for it. If such an allocation exists and can be found, *T* comes into the ready queue and becomes a ready task *RT(x1, y1, x2, y2, s, f)*. Coordinates *(x1, y1, x2, y2)* determine the position of the allocation. Time *s* represents start time and *f* represents the finish time of the task.

Fig.1 shows a 3D model of ready task. The horizontal direction reflects spatial requirement, and the vertical direction stands for temporal requirement. Parameter t_{config} is configuration time, t_{exe} is the execution time, t_{comm1} is the loading time of raw data, and t_{comm2} is the sending time of processed data to expected destination.

The task set is described by DAG as shown in Fig.2. In DAG scheduling, each node presents a task which executes a part of the application to be scheduled and is supposed to process some data.

Edges show the dependency among predecessor tasks and successor tasks. A predecessor task sends data to its successor task and determines the start of its successor task.

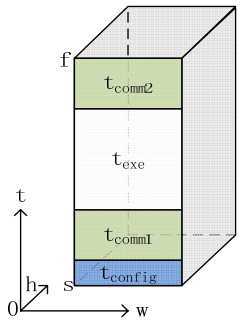


Figure 1. The task model

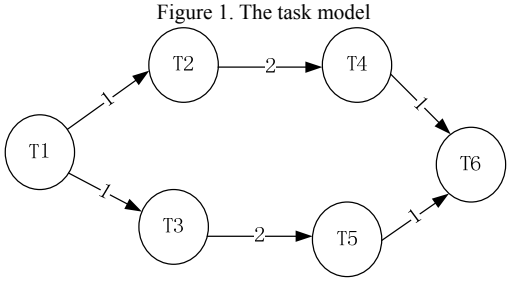


Figure 2. Task set described in DAG

A successor task receives data from its predecessor task and is driven by its predecessor task. The weight of each edge represents communication amount between dependent tasks. Communication amount is determined by task size and task execution time.

B. Communication Structure

In data intensive applications, data communication is an important aspect influencing the system performance [15][16]. Data communication can be reduced either by increasing communication throughput or decreasing the amount of data movement [17]. The former method is related to physical platform, so we focus on the latter one.

We adopt a new system model [18], the 2D area model with communication infrastructure as illustrated in Fig.3. This communication structure consists of communication channels, local buses, system buses and peripheral buses. In this model, the configuration area is first partitioned into slots, and then each slot is further partitioned into RCUs which are the basic units to run the logic function. Fig.3 also illustrates an example of communication among tasks and one peripheral. Task *T1* and *T2* are directly connected via the local buses through the channel; *T1* is connected with *T3* simply by local bus; *T3* and *T4* are connected to the same system bus for data exchange; *T4* is connected to the peripheral bus for communicating with external devices.

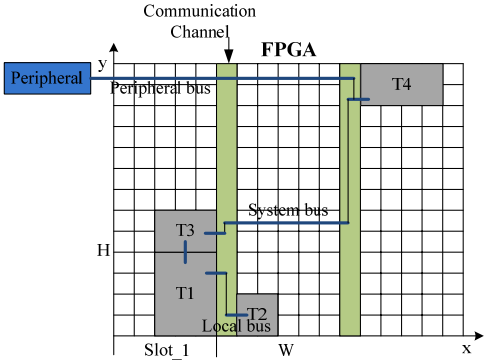


Figure 3. The 2D area model with communication structure

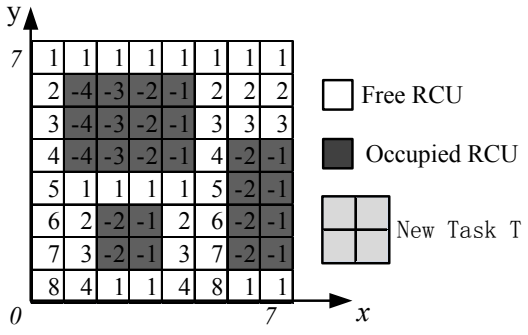


Figure 4. The mapping matrix

The communication has the following properties:

1. local communication has zero costs,

2. communication cost caused by either system bus or peripheral bus is in direct proportion to the distance between communication partners,

3. communication can be performed concurrently,

4. communication network is fully connected.

All these communication properties should be considered to avoid placing communicating tasks far apart, which would lead to high latency and cost.

IV. The Scheduling Algorithm

Many research efforts have been done to improve the computation density when allocating tasks onto an FPGA device, including bin-packing based algorithms (e.g. [4], [8], [19]) and adjacency based heuristics (e.g. [10], [20],). In this work we focus on the latter as it outperforms the former [21]. In the previous research, Yi Lu et al. [18] proposed a Communication Aware online task Scheduling Algorithm (CASA) calling the modified Flow Scanning (mFS) to manage resource. The execution time of mFS based on MER (Maximal Empty Rectangle)[21] rises sharply with the increase of tasks. So in accord with the authors' conclusion, the mFS is not suitable for the scheduling of large amount tasks. Therefore, we adopt other techniques to achieve efficiently scheduling and our algorithm includes three main parts: resource management, resource allocation and communication-aware scheduling.

A. Resource Management

Our resource management is based on the correlation matrix [22]. As shown in Fig.4, we establish a one-to-one mapping of a two-dimensional matrix $R[W][H]$ and every RCU of RLD. Parameter W is the width of RLD, H is Height. Positive weights represent free RCUs and negative weights represent the occupied. Weights of free RCUs are in descending order in y-axis direction with the minimal weight 1 and weights of occupied RCUs are in ascending order in x-axis direction with the maximal weight -1. This matrix clearly describes the size of the device and efficiently records the runtime placement of tasks.

To further improve the efficiency of resource management, we proposed a SA (Segmentation Algorithm) which improves the algorithms proposed in [6] and [11]. SA can find all available allocations for new arriving tasks and make it easier for allocation strategy to locate tasks quickly. SA skips the shielding sub-area during scanning by segmenting areas. During the horizontal scanning, SA directly skips the area with negative weights. During the vertical scanning, SA gets the difference between the boundaries' weights of the target area and check if the difference is less than the height of task to decide whether to save this area or not. The pseudocode of SA is shown in Algorithm1.

```
Algorithm1: SA(Task, x1, y1, x2, y2)
1   if x2 < x1 + Task.w then
2      return
3   end if
4   x ← x1
5   y ← y1
6   while y ≤ y2 and y ≤ H – Task.h + 1 do
7      while x < x2 – Task.w do
8         if isValidArea(x, y, Task, &p) then
9            save(x, y)
10           x++
11        else
12           jump(x, y) ← getJump(p, y)
13           if jump.y > y2 then
14              jump.y ← y2
15           end if
16           SA(Task, x1, y + 1, p - 1, jump.y)
17           SA(Task, jump.x + 1, y, x2, jump.y)
18           y ← jump.y
19        end if
20     end while
21     y++
22     x ← x1
23  end while
```

SA begins scanning from the bottom left point *(x1, y1)* and firstly scans x-axis then y-axis. Coordinate *(x2, y2)* is the top right point of candidate area. Function *isValidArea(x, y, Task, &p)* is called to check out if there is occupied RCU within the needed row and column. If there is no occupied RCU, function *save(x, y)* will be called to save this point as a candidate placement. Otherwise, SA will skip this area by getting the top right point of these occupied RCUs from function *getJump(p, y)*. These occupied RCUs will be skipped without scanning and thus the target area will be divided into three subareas: the left subarea, the right subarea and the top subarea. Then SA will recursively scan the entire reconfigurable resource area in left, right, and top sequential order.

The time complexity of SA algorithm is O(W×H) according to all RCUs in the worst case, while scanning is unnecessary in the best case. This is generally better than the existing resource management strategies.

B. Resource Allocation

According to the fragmentation metric proposed by Septién [22], scheduling algorithm which makes new arriving tasks have maximum adjacent edges will make the free area more like quadrate and more conducive to schedule the subsequent task.

We proposed a Maximum Adjacency Edges (MAE) algorithm which improved the BV (Boundary Value)

strategy [11]. MAE aims at placing task at the position with the largest adjacency in x and y directions with previously running tasks or RLD's boundaries. MAE algorithm has three basic steps. When a task arrives, firstly, SA is called to find an available allocation. Then this allocation is checked and the value of the adjacent edges is computed and maintained until a larger adjacent value is got. Finally, the candidate allocation with the maximum adjacent value will be chosen as the placement of the task which will be scheduled onto the RLD at its starting time.

The process of calculating the adjacent value is the essential part of MAE algorithm. The pseudocode of getting adjacent value of one candidate area is shown in algorithm2.

Algorithm 2: getAdjValue(x1, y1, x2, y2):

```
1   count ← 0
2   for x from x1 to x2 do
3      if RCU(x, y1 - 1) < 0 then
4         count ← count + 1
5      end if
6      if RCU(x, y2 + 1) < 0 then
7         count ← count + 1
8      end if
9   end for
10   for y from y1 to y2 do
11      if RCU(x1 - 1, y) < 0 then
12         count ← count + 1
13      end if
14      if RCU(x2 + 1, y) < 0 then
15         count ← count + 1
16      end if
17   end for
18   return count
```

MAE algorithm scans the weights of RCUs surrounding candidate area *(x1, y1, x2, y2)*. If some RCU's weight is negative, this RCU is occupied and the adjacent value increases.

In the fragmentation metric, authors assume the perimeter of all free resource is P, the total area is A, A_Q is the area of a perfect square with the same perimeter P, computed as:

$$A_Q = (P/4)^2 \tag{1}$$

Then the relevant quadrature is:

$$Q = A / A_Q \tag{2}$$

And then fragmentation is:

$$F = 1 - Q \tag{3}$$

It can be seen that the longer perimeter gives a higher fragmentation. In MAE algorithm, the external edges of a new arriving task $T(w, h)$ is $2\times(w + h)$ and the adjacent value of a valid allocation for $T(w, h)$ is AE.

After task T is allocated, the layout of RLD is changed and the total perimeter of the entire free area becomes P'. P' can be calculated by:

$$P' = P + 2\times(w+h) - 2\times AE \tag{4}$$

So in order to decrease P', we should make AE as large as possible. Conclusively, a larger adjacent value provides a lower fragmentation, i.e. MAE algorithm is in accord with fragmentation metric. Fig. 5 shows an instance to illustrate MAE algorithm. When new task *T(2, 2)* comes, there are three tasks running on RLD, P is 58, and A is 38. There are three candidate areas for T, and each area has more than one appropriate placement. Take placement $A1$ and $A2$ for example:

$A1$: AE is 6, P' is 50, F is 0.7824.

$A2$: AE is 6, P' is 54, F is 0.8134.

The result obviously shows that compared with putting task T at $A2$, putting T at $A1$ results in more adjacent edges and smaller fragmentation.

C. Communication Coefficient

Based on the communication structure, we abstract communication channel into a special task occupying $1\times H$ RCUs, as shown in Fig.6. The original fragmentation is 0. After placing a task *T(3, 3)* in Fig. 6(a) and Fig. 6(b), the fragmentation is unchanged. However, the communication properties cannot be reflected in this case. To solve this problem, we use a communication coefficient α to emphasize the communication requirement and develop MAE to communication-aware MAE (CA-MAE) scheduling algorithm. This coefficient will optimize the tasks placement by reducing the distance between communication partners. The communication coefficient α of each is

$$\alpha = \frac{2n - i}{n} (1 \le i \le n) \tag{5}$$

In equation (5), parameter n is the width of a slot, and i is the distance between the RCU and communication channel. Multiplying the adjacent value of y direction by α, we can get a different result.

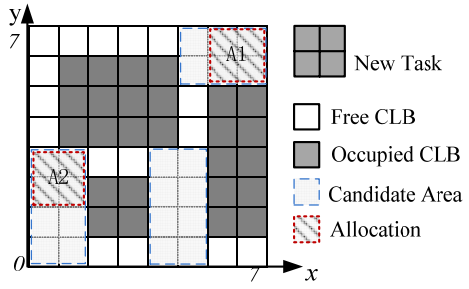


Figure.5. An instance of MAE

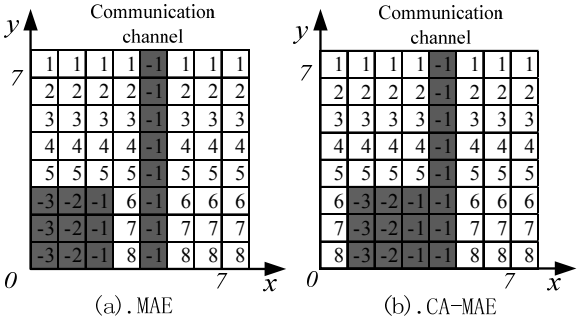


Figure 6. The CA-MAE scheduling algorithm

The maximum adjacent edges in Fig.6(a) are still 6. But in Fig.6(b), the adjacent edges of task *T*'s right side are adjacent to communication channel. Since α is (2×4-1)/4 = 1.75, this value becomes 5.25 after multiplying α. Adding the adjacent value of task *T*'s bottom, the maximum adjacent edges is 8.25.

It can be seen that allocating task closer to the channel will reduce both its occupation of bus and communication time. In the scheduling process, communication channel is like a magnet, attracting tasks to be placed closer to it, thus the RLD's layout is more compact. In DAG scheduling, a task cannot begin execution until all its inputs have arrived and no output is available until the computation has finished. By reducing communication time between dependent tasks, CA-MAE algorithm accelerates the whole computing process of task set.

V. Experimental Evaluation

The ideal performance assessment should consider the practical application, however, the actual system supporting reconfigurable computing does not exist yet. Researchers have no choice but to implement the algorithm by simulating. We simulated our algorithm by generating randomly simulation tasks, as Zhou [11] and Lu [18] did.

In our simulations, we considered communication as one of the basic constraints and compared MAE, CASA, FF (First Fit) [4] with CA-MAE. Performance of these strategies is evaluated by the utilization of RLD, fragmentation and ACT (Application Completion Time). We assumed that the input edge of tasks could be placed next to the output edge of the predecessors, as the Snake algorithm [7] does.

The target RLD contains 48 (columns) × 16 (rows) RCUs and the different task sets are randomly generated, similar to others in multitasking environment. Task size is uniformly distributed in [4~30] RCUs. The width is each task is not larger then the width of one slot. And each task is only allowed to be placed the neighboring RCU in the same slot.

Task execution time is between 3 and 20ms. The configuration time of per RCU is about 0.34ms. The load ratio [11] is adopted to determine the arrival frequency of tasks which has a great influence on the scheduling success rate. The t_{comm} is in direct proportion to communication amount and the distance between two communication partners. The t_{comm} is computed in the following equation:

$$t_{comm} = \beta \times size \times t_{exe} \times L \tag{6}$$

Since parameter L is the distance between two communication partners which ranges from 1 to W, we set factor β as the logarithm of L to make communication time compatible to configuration time.

All evaluated algorithms are programmed in C++ and simulated in Windows XP with Pentium(R) Dual-Core CPU @3.2GHz.

A. Comparison of ACT

In the simulation, each algorithm processed 1000 applications made of random tasks. Fig.7 shows that CA-MAE outperforms others in ACT. Since ACT is highly affected by the communication time, and CA-MAE reduces communication cost during scheduling and further reduces the ACT. With the increase of the number of tasks, the free resources of RLD and the candidate allocations decrease, then the advantage of CA-MAE is no longer so obvious.

B. Comparison of Fragmentation

In Fig.8, it can be seen that FF algorithm always has the higher fragmentation and CA-MAE algorithm has the lower one. This is because FF algorithm places a task at the first position it found regardless of its consequent influence. According to paper [21], CA-MAE makes the free area more quadrate and is more conducive to schedule the subsequent tasks.

C. Comparison of Utilization

The reuse of reconfiguration resources is pretty important for the reconfigurable system and the utilization is closely related to allocation strategy. For the limited RCU resources, there is an upper limit of the actual utilization which is decided by the following equation.

$$U = \frac{\sum_{T_i} w_i \times h_i \times t_{ei}}{W \times H \times t_{fl}} \tag{7}$$

Parameter t_{ei} consists of t_{config}, t_{exe} and t_{comm}. The t_{fl} is finish time of the last task. This equation illustrates that the utilization is the ratio of occupied resources to the total resources.

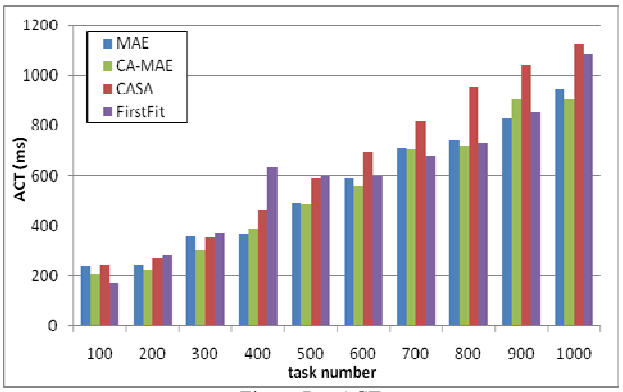


Figure 7. ACT

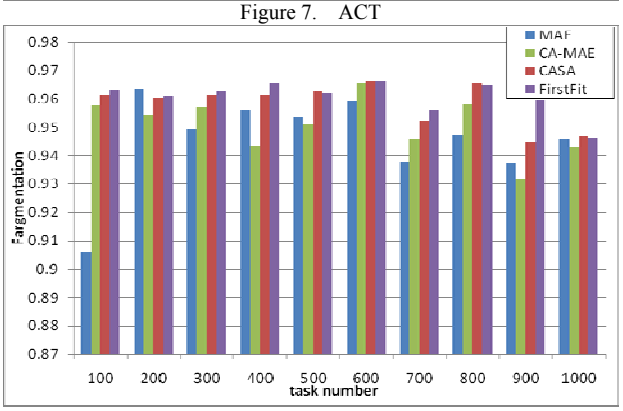


Figure 8. Fragmentation

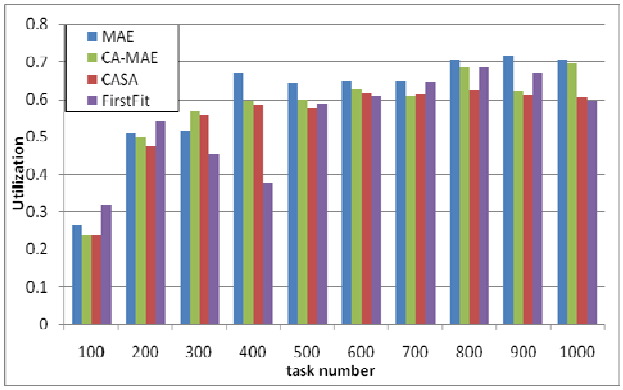


Figure 9. Utilization

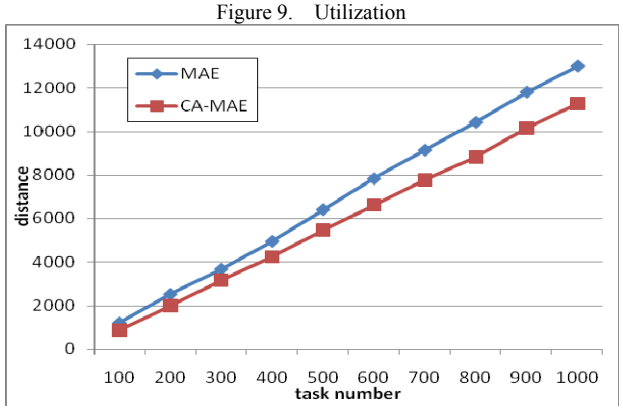


Figure 10. Evaluation of algorithms

As shown in Fig.9, MAE outperforms other algorithms. Since the communication partners in MAE may have a longer communication distance and a longer t_{comm} which extends the denominator in equation (7).

As illustrated in Fig.10, we compared MAE with CA-MAE. By taking data communication into consideration, CA-MAE reduces distance between communication partners. The result shows that CA-MAE decreases 17% communication distance on average.

VI. Conclusion

The distance between communication partners directly impacts the system performance, as longer distance leads to higher latency and communication load. In this work, we presented SA algorithm to efficiently manage the reconfigurable resource and developed a 2D reconfigurable model with full consideration of communication resource. Based on this model, we proposed a CA-MAE algorithm that reduces communication distance by considering communication properties during scheduling. CA-MAE algorithm optimizes the placement of MAE. The experimental results show that CA-MAE can reduce communication cost by 17% and reduce the application completion time by 21%.

Acknowledgment

This work was supported by the National High-Tech Research and Development Plan of China under Grant No.2012AA01A301-01, the National Natural Science Foundation of China under Grant No.61173036 and No.61300037 and Hunan Provincial Natural Science Foundation of China under Grant No. 12JJ4057.

References

[1] M. Huang, VK. Narayana, H. Simmler, O. Serres, and T. EI-Ghazawi, "Reconfiguration and communication-aware task scheduling for high performance reconfigurable computing," *ACM Transaction on Reconfigurable Technology and Systems*, vol 3, 2010.

[2] K. Pocek, R. Tessier R, A. DeHon. "Birth and adolescence of reconfigurable computing: A survey of the first 20 years of field-programmable custom computing machines," *Highlights of the First Twenty Years of the IEEE International Symposium on Field-Programmable Custom Computing Machines*, pp. 3-19, 2013.

[3] X. Wang, L. Dong, L. Gui, et al. "Research on the Cost of Open Nested Transaction in the Open Reconfigurable Network," Journal of Software (1796217X), 2013, 8(6), pp. 1411-1418.

[4] K. Bazargan, R. Kastner, and M. Sarrafzadeh, "Fast template placement for reconfigurable computing systems," *IEEE Design and Test of Computers*, vol. 17, pp. 68-83, January 2000.

[5] C. Steiger, H. Walder, and M. Platzner, "Heuristics for online scheduling real-time tasks to partially reconfigurable devices," *Field Programmable Logic and Application 2003*, Springer Berlin Heidelberg, vol. 2778, pp. 575-584, September 2003.

[6] T. Marconi, Y. Lu, K. Bertels, and G. Gaydadjiev, "Online hardware task scheduling and placement algorithm on partially reconfigurable devices," *Reconfigurable Computing: Architecture, Tools and Applications. UK*, vol. 4943, pp. 306-311, March 2008.

[7] X. Iturbe, K. Benkrid, C. Hong, A. Ebrahim, T. Arslan, and I. Martinez, "Runtime scheduling, allocation, and execution of real-time hardware tasks onto Xilinx FPGAs subject to fault occurrence," *International Journal of Reconfigurable Computing*, vol. 2013, 2013.

[8] J. Tabero, J. Septién, H. Mecha, and D. Mozos, "Task placement heuristic based on 3D-adjacency and look-ahead in reconfigurable systems," *Proceedings of the 2006 Asia and South Pacific Design Automation Conference*, Japan, pp. 396-401, January 2006.

[9] J. Valero, J. Septién, D. Mozos, and H. Mecha, "3D FPGA resource management and fragmentation metric for hardware multitasking," *IEEE International Symposium on Parallel & Distributed Processing, 2009*, Italy, pp.1-7, May 2009.

[10] T. Marconi, Y. Lu, K. Bertels, and G. Gaydadjiev, "3D compaction: a novel blocking-aware algorithm for online hardware task scheduling and placement on 2D partially reconfigurable devices," *Reconfigurable Computing: Architecture, Tools and Applications. Thailand*, vol.5992, pp. 194-206, March 2010.

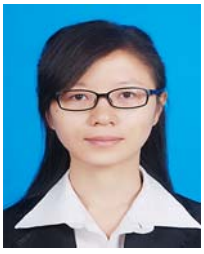
[11] X. Zhou, L. Liang, X. Huang, and P. Cheng, "On-Line scheduling and placement of real-time tasks for reconfigurable computing system," *IEEE International Conference on Field Programmable Technology 2006*, Thailand, pp. 57-64, December 2006.

[12] J. Teller, F. Ozguner, and R. Ewing, "Scheduling task graphs on heterogeneous multiprocessors with reconfigurable hardware," *Parallel Processing - Workshops*, USA, pp. 17-24, September 2008.

[13] G. R. Morris, K. H. Abed. "Mapping Floating-Point Kernels onto High Performance Reconfigurable Computers," *Journal of Computers*, vol. 8, No. 4, pp. 859-873, April 2013,.

[14] M. C. Eddin. "Towards a Taxonomy of Dynamic Reconfiguration Approaches," Journal of Software

(1796217X), 2013, 8(9), pp. 2202-2207.
[15] C. Pham-Quoc, Z. Al-Ars, and K. Bertels, "A heuristic-based communication-aware hardware optimization approach in heterogeneous multicore systems," *Reconfigurable Computing and FPGAs*, Mexico, pp. 1-6, December 2012.
[16] Z. Lin, L. Wang, J. Zhou, et al. "The Research on Fatigue Driving Detection Algorithm," Journal of Software (1796217X), 2013, 8(9), pp. 2272-2279.
[17] P. Mahr, and C. Bobda, "Reducing communication costs on dynamic networks-on-chip through runtime relocation of tasks," *Rapid System Prototyping*, Finland, pp. 177-182, October 2012.
[18] Y. Lu, T. Marconi, K. Bertels, and G. Gaydadjiev, "A communication aware online task scheduling algorithm for FPGA-based partially reconfigurable systems," *18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines 2010* , USA, pp. 65-68, May, 2010.
[19] H. Han, et al. "Efficient Algorithm for Hardware/Software Partitioning and Scheduling on MPSoC," *Journal of Computers,* vol. 8, No. 1, January 2013.
[20] X. Iturbe, K. Benkrid, T. Arslan, C. Hong, and I. Martinez, "Empty resource compaction algorithms for real-time hardware tasks placement on partially reconfigurable FPGAs subject to fault occurrence," R*econfigurable Computing and FPGAs*, Mexico, pp. 27-34, November 2011.
[21] M. Handa, and R Vemuri, "An efficient algorithm for finding empty space for online FPGA placement," *Design Automation Conference*, USA, pp. 960-965, June 2004.
[22] J. Septién, D, Mozos, H. Mecha, J. Tabero, and M. A. G. de Dios, "Perimeter quadrature-based metric for estimating FPGA fragmentation in 2D HW multitasking," *IEEE International Symposium on Parallel and Distributed Processing 2008,* USA, pp. 1-8, April 2008.



Yingying Sheng, born in China, in 1988. Got bachelor degree in information security from Hunan University, Changsha, China, in 2011.

She is a postgraduate student of college of Information Science and Engineering, Hunan University. Her research interests include computer architecture and embedded system.

Yan Liu, born in China, in 1979. Got the PhD in computer science from Hunan University, China, in 2010.

He is an assistant professor of college of Information Science and Engineering, Hunan University. His research interests include computer architecture, multicore scheduling, distributed computing systems.

Renfa Li, born in China, in 1957. Got PhD degree in electric engineering from Huazhong University of Science and Technology, in 2003. Got master degree in electromagnetic computing from Tianjin University, in 1987. Got bachelor degree in electrical engineering and automation from Tianjin University in 1982.

He is a professor and PhD supervisor of Hunan University and has published over 60 scholarly papers in journals, book chapters and international conferences and acted as editor of a dozen books in recent years. His main research interests include CPS, embedded systems and wireless sensor networks.

Prof. Li is a councillor of China Computer Federation (CCF), a senior member of the Institute of Electronic and Electrical Engineers, a senior member of the Association for Computing Machinery. Prof. Li has been a chief investigator on research grants from different sources including Natural Science Foundation of China (NSFC) and others at state or provincial level.