Miss Ratio Improvement For Real-Time Applications Using Fragmentation-Aware Placement

Ahmed Abou ElFarag abouelfarag@aast.edu
Hatem M. El-Boghdadi helboghdadi@eng.cu.edu.eg
Samir I. Shaheen sshaheen@ieee.org

Department of Computer Engineering
Cairo University, Giza, EGYPT

Abstract

Partially reconfigurable Field-Programmable Gate Arrays (FPGAs) allow parts of the chip to be configured at run-time where each part could hold an independent task. Online placement of these tasks result in area fragmentation leading to poor utilization of chip resources.

In this paper, we propose a new metric for measuring area fragmentation. The new fragmentation metric gives an indication to the continuity of the occupied (or free) space and not the amount of occupied space. We show how this metric can be extended for multi-dimensional structures. We also show how this metric can be computed efficiently at run time. Next we use this measure during online placement of tasks on FPGAs, such that the chip fragmentation is reduced. Our results show improvement of chip utilization when using this fragmentation aware placement method over other placement methods with well known Bottom Left First Fit, and Best Fit placement strategies. In real time environment, we achieve an improvement in miss ratio when using the fragmentation aware placement over the bottom left placement strategy.

1 Introduction

Partially reconfigurable FPGA aims to allow part of the device to be reconfigured while another part is still performing active computation. A device that has this property can accommodate number of applications at the same time where each application can be composed of number of tasks. To take full advantage of the partially reconfigurable fabric, good utilization of chip resources is a must [3]. Uncareful placement of incoming tasks cause portions of chip area to be wasted because they are too small to hold another incoming task. These small portions are called fragments similar to the fragments in traditional memory systems, and they could represent significant percentage of chip area [7]. Consequently, area fragmentation is one of the biggest obstacles of obtaining good utilization of chip resources.

In an online placement systems, due to dynamic addition and deletion of rectangular tasks, the empty area of the FPGA become highly fragmented and thus FPGA area cannot be utilized efficiently. Figure 1 shows one such example where tasks $T1$ and $T2$ cause fragmentation of the FPGA area. As a result, a task $T3$ may not be placed on the FPGA surface even if total empty area on the FPGA is more than the area of the task $T3$.

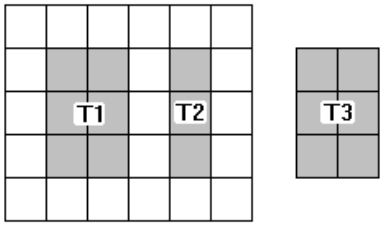


Figure 1. Fragmentation of FPGA area.

In this paper, we propose a new metric for measuring area fragmentation. The new fragmentation metric gives an indication to the continuity of the occupied (or free) space on the reconfigurable chip and not the amount of occupied (or free) space. High area fragmentation indicates the scattering of occupied portions on the chip and hence could cause low chip utilization. On the other hand, low area fragmentation enables incoming tasks to be placed enhancing the chip utilization. This measure can be used in monitoring the chip area and select the best empty area to place the new task and thus reducing the total chip area fragmenta-

1-4244-0910-1/07/$20.00

tion. We extend this metric for multi-dimensional structures such as hyper-cubes. We also show how this metric can be computed efficiently at run time. Next we use this measure during online placement of tasks, such that the chip fragmentation is reduced. We also use this metric in real time systems to reduce the miss ratio. Our results regarding chip utilization show 5% to 10% improvement for this fragmentation aware placement method over placement with well known First Fit, Best Fit and Bottom Left placement strategies. In real time applications, the miss ratio is improved by up to 7%.

In Section 2, we discuss some of the previous work in the fragmentation quantification. In Section 3, we clarify the system model. Section 4 describes the new fragmentation metric and its extension to multi-dimensional structures. Section 5 introduces a methodology to calculate the fragmentation metric. Section 6 shows the results of using the metric during online placement and its effect on area utilization. In Section 7 we summarize our results and make some concluding remarks.

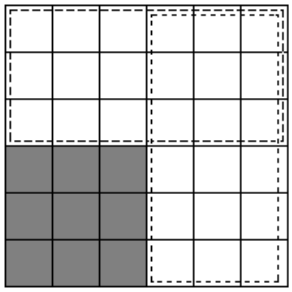
2 Related Work

In this paper, a highly fragmented chip is the one, which contains large number of holes separated by some occupied cells. We focus on the external fragmentation [5], which is the fragmentation of area resources outside the rectangular boundary of the tasks. A task may be denied placement if enough contiguous resources are not available for its rectangular area. We attempt to quantify the fragmentation of the area resources by calculating Fragmentation Factor or fragmentation metric.

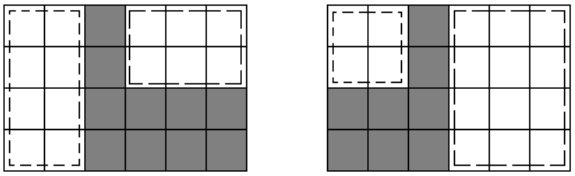
Two main directions were followed to enhance area utilization of the chip. One allowed task pre-emption (i.e. stopping the active tasks, preserving them or moving them to other portions, and reconfigure the new portion). Diessel and ElGindy [1] presented a heuristic for task compaction that is one dimensional and order preserving. The technique attempts to compact running tasks in a certain direction if the incoming task cannot be allocated (placed) on the chip area. This compaction, in turn, enables incoming tasks to be placed leading to a better utilization.

The other direction was through fragmentation aware placement techniques in which careful placement is done as to reduce the area fragmentation allowing more space for future task allocation.

A reliable measure of fragmentation is needed to decide where to put the new task. All empty places have to be tested to select one that causes lower fragmentation. Thus, this placement strategy takes into consideration the fragmentation measure value in place selection. This method is followed in this paper and called Fragmentation Aware Placement strategy.



(a) Overlapped area.



(b) Fragmentation factor ambiguity.

Figure 2. Examples of Fragmentation calculations.

Wigley *et al.* [9] used square of shorter side (which they call characteristic dimension) of an empty rectangle in the reconfigurable area as a measure of fragmentation. They calculated distribution of characteristic dimension and used the mean of that distribution as a fragmentation metric. It is clear that different distributions (fragmentation states) may have the same mean. In other words, the mean of the distribution does not uniquely identify how much the chip is fragmented.

Walder *et al.* [8] used fragmentation grade to quantify the fragmentation. They found all the empty rectangles required to cover empty area on the FPGA. If a_i is the area of i^{th} empty rectangle then fragmentation grade is calculated as $F = 1 - \frac{\sqrt{\sum {a_i}^2}}{\sum a_i}$. The empty rectangles used in the previous equation may overlap, and so empty areas may participate in the fragmentation grade more than once. This may give inaccurate results. As shown in Figure 2(a), the fragmentation grade used the same area twice.

Ejnioui and DeMara [2] used fragmentation metric to quantify the degree of scattering of the holes across the chip area as the chip fragmentation. $f_i = \frac{a_i}{A}$, and $F = 1 - \prod_i f_i$ where a_i is the area of the i^{th} empty rectangle and A is the chip area. They found the maximum empty rectangles required to cover empty area on the FPGA, and calculate the fragmentation factor. The problem of empty area overlapping still exists here. Also two different fragmentation states may give the same fragmentation factor, as shown in Figure 2(b).

All the above methods give absolute fragmentation metrics. They take into consideration only the distribution of the reconfigurable cells. Another trend is the relative fragmentation metric which consider the state of the reconfigurable cells based on the sizes of the incoming tasks.

As an example of relative fragmentation metric, M. Handa and R. Vemuri [4], base their calculations on the average size of the tasks. They used number of empty cells in the vertical and horizontal vicinity of an empty cell, and if this number is greater than or equal to double the average size of incoming tasks, the fragmentation contribution of this cell is 0. The total fragmentation is calculated by the sum of vertical and horizontal fragmentation of all empty cells in the reconfugurable chip area.

Relative metrics are not sensitive to the size of the incoming task at certain time, so it is possible to obtain two different fragmentation measures at different times for the same fragmentation state of the chip while the size of the incoming tasks may be highly variable.

As a result, an absolute metric is more accurate. It is not required to give any indication of how full or empty the chip is. It should instead give an indication of the degree of scattering of the holes (or empty areas) across the chip area.

In this paper, we introduce a new measure for area fragmentation that is absolute. The new measure gives an indication for the continuity of the occupied (or free) cells within the FPGA independent of the incoming tasks' sizes.

3 System Model

In this section, we present the tasks and system model used in this paper for a partial reconfigurable system.

An $H \times W$ partially reconfigurable FPGA chip consists of H rows and W columns. The lower left corner is the chip origin. Part of the chip can be configured without affecting the rest of the chip. The FPGA is partially reconfigurable with reconfiguration time proportional to the number of cells being reconfigured since cells are configured sequentially. Configuration delay is the time to configure a cell and its associated routing resources.

The system assumes that tasks arrive online, queued and placed in arrival order. As long as free cells are available in the FPGA area, the server continues to service incoming tasks by placing each one on an unoccupied area of the FPGA chip. If a task on the top of the queue is denied placement due to non-availability of contiguous resources (free cells), the placement queue is stalled till contiguous sufficient empty space exists.

Tasks are non-preemptive. Once they are placed at any free area in the reconfigurable chip area, and stay at this place till finishing the execution. Task parameters are not known in advance. These task parameters are defined as: for a task t_i , $t_i = (h_i, w_i, a_i, s_i, d_i, x_i, y_i)$, h_i and w_i represent its height and width respectively and they are measured in number of cells, a_i, s_i and d_i are the tasks arrival times, the tasks service times, and the tasks deadline times respectively. The rectangular area assigned to the task is presented by its lower left corner (x_i, y_i) where x_i: row number, and y_i: column number. These characteristics reflect a general-purpose computing system. The size of the task, its arrival time, service time, and deadline time are uniformly distributed in a predefined region and a-priori unknown.

Formally, a task t_i arrives at time a_i , starts to execute at time s_i , and finishes at f_i . Thus, the task's waiting time is given by $wait(t_i) = s_i - a_i$ and the response time is given by $resp(t_i) = f_i - a_i$. The allocation time of a task, $alloc(t_i)$ is the time the task is waiting at the top of the waiting queue till it finds a place on the reconfigurable area.

4 Fragmentation Metric

In this section we introduce a new fragmentation metric for multi-dimensional structures. Our metric does not tie itself to a specific application. Hence, it can be used or tailored to the application at hand. Assume that cells are the smallest units of area resources. Used cells are referred as occupied cell and unused calls called empty. We start with a metric for one-dimensional structures, then we extend it to two-dimensional structures and higher.

4.1 One-Dimensional Fragmentation Metric

In this section we introduce a new metric for one-dimensional structures. We first introduce some definitions.

Consider a one-dimensional structure, S, that is composed of L cells. Each cell could be in either state (occupied or empty). We call this structure a cell stream.

Definition 1 A cell stream S of length L is a group of L consecutive cells.

Remark: A cell stream represents a group of consecutive cells regardless the state of each cell.

Definition 2 A cell sequence Q of length K is a group of K consecutive empty cells.

In Figure 3(a), there exist two cell sequences of lengths $2, 3$ respectively. While in Figure 3(b), there exist three cell sequences of lengths $1, 4, 5$ respectively. Let $c(S)$ be the number of cell sequences in the cell stream S. Let $Q_S(x)$ denote the x^{th} cell sequence in the stream S and $\ell(Q_S(x))$ be its length ($0 \leq x \leq c(S)$).

Lemma 1 *The number of cell sequences, $c(S)$, in a cell stream S of length L is $\leq \lceil \frac{L}{2} \rceil$.*

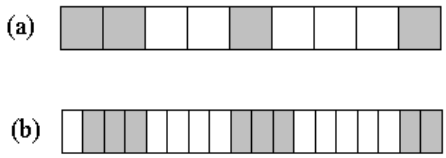


Figure 3. Illustartion of one-dimensional framentation metric.

Proof: The maximum number of cell streams happens when each cell stream is of length one. This occurs when the cell sequence has one empty cell followed by one occupied cell and so on. Then the number of empty cells represents the number of cell sequences which is $\frac{L}{2}$ for a cell stream of length L. ■

Let $\hat{Q_S} = \{Q_S(x) : 0 \leq x \leq c(S)\}$ be the set of cell sequences in cell stream, S of length L. Clearly, the maximum number of cell sequences $|\hat{Q_S}|$ is $\leq \lceil \frac{L}{2} \rceil$.

Our metric measures the continuity of the occupied area within the cell stream and does not measure how many cells is empty. Thus, it does not tie itself to a specific problem, rather, it is a universal metric and could be used in a different setting. Measuring the continuity of the occupied cells in a cell stream translates to measuring how many cell sequences within the cell stream. The existence of many cell sequences with small lengths is an indication to high fragmentation. We build on this observation.

Let F_S denote the fragmentation metric of the cell stream, S. As the number of cell sequences increase, the fragmentation of S increases. The length of each cell sequence is another important factor in computing the fragmentation of S. As the length of each cell sequence decreases, the cell sequence has more contribution to the fragmentation metric.

A cell sequence contributes to F_S with a value of $\frac{1}{\ell(Q_S(x))}$. Then, the fragmentation metric,

$$F_{1-d} = \sum_{x=0}^{|\hat{Q_S}|-1} \frac{1}{\ell(Q_S(x))}. \quad (1)$$

In Figure 3(a) and (b), $F_{1-d} = \frac{1}{2} + \frac{1}{3} = \frac{5}{6}$ and $F_{1-d} = \frac{1}{1} + \frac{1}{4} + \frac{1}{5} = \frac{29}{20}$ respectively.

4.2 Two-Dimensional Fragmentation Metric

Here, we extend the metric introduced in the previous section to two-dimensional structures. Consider a two-dimensional structure (array), A, composed of $H \times W$ cells. Each cell could be in either state (occupied or empty). This makes a good matching with the FPGA model presented in section 3.

Let $\hat{Q}_{R(i)} = \{Q_{R(i)}(x) : 0 \leq x \leq c(R(i))\}$ be the set of cell sequences within row i. Similarly, let $\hat{Q}_{C(j)} = \{Q_{C(j)}(y) : 0 \leq y \leq c(C(j))\}$ be the set of cell sequences within column j, where $0 \leq i \leq H-1$ and $0 \leq j \leq W-1$. Clearly, the maximum number of cell sequences $|\hat{Q}_{R(i)}|$ is $\leq \lceil \frac{W}{2} \rceil$ where $0 \leq i \leq H-1$. Also, the maximum number of cell sequences $|\hat{Q}_{C(j)}|$ is $\leq \lceil \frac{H}{2} \rceil$ where $0 \leq j \leq W-1$.

Each row and column is a one dimensional structure and the result in section 4.1 could be applied. Each row (column) contributes to the area fragmentation metric . Again, we here measure the continuity of the occupied area within the array and does not measure how much area is empty.

Let $F_{R(i)}$ and $F_{C(j)}$ denote the the contribution of row i and column j to the fragmentation metric, F_{2-d} of the chip. A cell sequence $Q_{R(i)}$ contributes to $F_{R(i)}$ with a value of $\frac{1}{\ell(QR_i(x))}$. Then, $F_{R(i)} = \sum_{x=0}^{|\hat{Q}_{R(i)}|-1} \frac{1}{\ell(Q_{R(i)}(x))}$.

In an FPGA setting, each row i contributes to the total area fragmentation metric with a value of $F_{R(i)}$. Let F_R denote the contribution of all rows in the chip. Then $F_R = \sum_{i=0}^{H-1} F_{R(i)}$.

Similarly, each column contributes to the total metric with a value of $F_{C(j)} = \sum_{y=0}^{|\hat{Q}_{C(j)}|-1} \frac{1}{\ell(Q_{C(j)}(y))}$. Let F_C denote the contribution of all columns in the chip. Then $F_C = \sum_{j=0}^{W-1} F_{C(j)}$. Finally, the fragmentation metric , F_{2-d}, can be calculated as $F_{2-d} = F_R + F_C$

$$F_{2-d} = \sum_{i=0}^{H-1} \sum_{x=0}^{|\hat{Q}_{R(i)}|-1} \frac{1}{\ell(Q_{R(i)}(x))} + \sum_{j=0}^{W-1} \sum_{y=0}^{|\hat{Q}_{C(j)}|-1} \frac{1}{\ell(Q_{C(j)}(y))}. \quad (2)$$

In Section 5 we use this metric in online placement of tasks on FPGAs.

4.3 Higher-Dimensional Fragmentation Metric

In this section we extend our results to n-dimensional structures such as hyper-cubes. Consider an n-dimensional array of cells $m_1 \times m_2 ... \times m_n$. Let $d_1, d_2, ..., d_n$ denote the n dimensions. The basic unit in this structure is the

basic logic block (cell). Each cell is either occupied or empty. We will consider each dimension and account for the contribution of this dimension to the fragmentation metric. We will extend the work done in section 4.2 to n dimensions. A cell in the array can be defined by determining the value in each dimension where the cell is located. For example, the cell $C(x_1, x_2, ..., x_n)$ where $0 \leq x_i \leq m_i - 1$ and $1 \leq i \leq n$ is the one located at x_1 along the d_1 dimension, x_2 along the d_2 dimension and so on. Let $F_{d_1}, F_{d_2}, ..., F_{d_n}$ denote the contribution of the $d_1, d_2, ..., d_n$ dimension respectively, to the fragmentation metric. Let $Q_{d_r(x_1,x_2,...,x_{r-1},x_{r+1},...,x_n)}(k)$ denote the k^{th} cell sequence in dimension d_r ,$1 \leq r \leq n$, and the values along the dimensions $d_1, d_2, ..., d_{r-1}, d_{r+1}, ..., d_n$ are $x_1, x_2, ..., x_{r-1}, x_{r+1}, ..., x_n$ respectively. Let $\hat{Q}_{d_r(x_1,x_2,...,x_{r-1},x_{r+1},...,x_n)}(k) = \{Q_{d_r(x_1,x_2,...,x_{r-1},x_{r+1},...,x_n)}(k) : 0 \leq k \leq c(d_r)\}$ be the set of cell sequences within dimension d_r Also let $\ell(Q_{d_r(x_1,x_2,...,x_{r-1},x_{r+1},...,x_n)}(k))$ denote this cell sequence's length. Extending the results in section 4.2, where $1 \leq r \leq n$, gives

$$F_{d_r} = \sum_{x_1=0}^{m_1-1} \sum_{x_2=0}^{m_2-1} ... \sum_{x_{r-1}=0}^{m_{x-1}-1} \sum_{x_{r+1}=0}^{m_{x+1}-1} ...$$

$$\sum_{x_n=0}^{m_n-1} \sum_{k=0}^{|\hat{Q}_{d_r}(k)|-1} \frac{1}{\ell(Q_{d_r(x_1,x_2,...,x_{r-1},x_{r+1},...,x_n)}(k))}.$$

The total fragmentation metric is the summation of each dimension contribution,

$$F_{n-d} = \sum_{r=1}^{n} F_{d_r}. \quad (3)$$

A direct application of this metric when $n = 3$ is to measure how scattered are the active nodes in hypercubes.

5 Methodology for Calculating the Fragmentation Metric

In this section, we present a runtime efficient algorithm for calculating the fragmentation metric at a given time. We maintain fragmentation information for rows (resp. columns) in the form of one dimensional array called R (resp. C). Fragmentation information is updated after every addition and deletion of a task to the FPGA. Initially, when the FPGA is empty, the value of each element in R and C is zero.

Figure 4 shows a chip with some running tasks. The arrays R and C that represents the chip area is shown in the figure along the columns and rows. Each element of R (resp. C) holds the number of contiguous empty cells

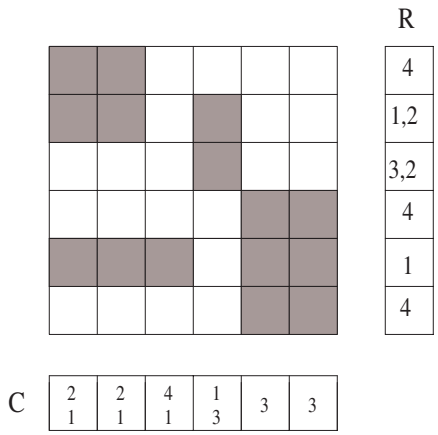


Figure 4. Calculating fragmentation metric.

in each row (resp. column). Element $R(i)$ (resp. $C(j)$) contains fragmentation information for row i (resp. column j). Each element in R (resp. C) may contain more than one value as a row (resp. column) may have more than one sequence of empty cells. Figure 4 shows an example of an FPGA device with 6 rows and 6 columns. In the figure, $R(0) = 4$ represents the number of contiguous empty cells in the first row. The first element of C, $C(0) = [2, 1]$, represents two distinct empty sequences of lengths 2 and 1 respectively.

The fragmentation metric for the chip, $F_{2-d} = F_R + F_C$ where, $F_R = \frac{1}{4} + \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{2} + \frac{1}{4} + \frac{1}{1} + \frac{1}{4}$ and $F_C = \frac{1}{2} + \frac{1}{1} + \frac{1}{2} + \frac{1}{1} + \frac{1}{4} + \frac{1}{1} + \frac{1}{1} + \frac{1}{3} + \frac{1}{3} + \frac{1}{3}$.

5.1 Addition and Deletion of Tasks

The fragmentation arrays R and C can be updated very efficiently after addition or deletion of tasks. When a task is added or deleted, it affects only the corresponding elements of R and C.

If a task is added to or removed from the chip and occupies rows r_i to r_j and columns c_i to c_j then the elements $R(r_i)$ to $R(r_j)$ will be changed accordingly to reflect the new status of these rows. The same effect will occur to elements $C(c_i)$ to $C(c_j)$.

6 Fragmentation Aware On-Line Placement

In this section, we use the two dimensional fragmentation metric discussed above to help in the Fragmentation Aware Placement (FAP) methodology. Again, the fragmentation metric measures the continuity of the occupied (free) area; i.e. chip states with lower fragmentation metric values

indicate more continuity of free space. We use this fact in choosing the place of the incoming tasks.

Once a task is on the top of the queue, the placement engine checks all the possible places for that task. The engine chooses the place that results in a lower fragmentation metric for the new state. This results in a more continuous space that would hopefully make a room for the next set of incoming tasks.

The fragmentation metric can be fast calculated using the arrays R and C as described in Section 5. The main contribution here is to find the best candidate place to put the task in as to reduce the fragmentation metric for the new state. The advantage of this method over the traditional placement methodologies will be shown next.

Based on various distributions of task sizes, inter-task arrival times, and task execution times, many runs are performed to observe the effect of these parameters on the system performance.

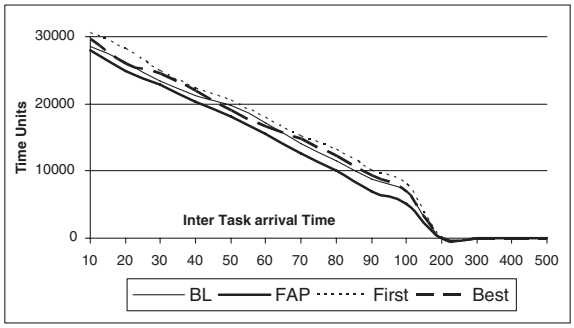
6.1 Experimental Results

A series of experiments was done to test the performance of the methods with synthetic task sets. For each experiment, a set of 1000 tasks characterized by 4 independently chosen uniformly distributed random variables was generated. Two of these variables, representing the length and width of the incoming task chosen randomly (uniformly distributed), were permitted to range from 1 to 32 cells each. A variable representing the tasks' service periods was generated randomly (uniformly distributed) between 1 and 1000 $timeunits$. The deadline time of a task is formed by adding a randome varible (uniformly distributed between 1 and 50) to the task's service time. The experiment was repeated with various inter task arrival period intervals uniformly distributed between 1 and (10,20, 30, 40, 50, 60, 70, 80, 90, 100, 200, 300, 400, and 500) $timeunits$ (tu).

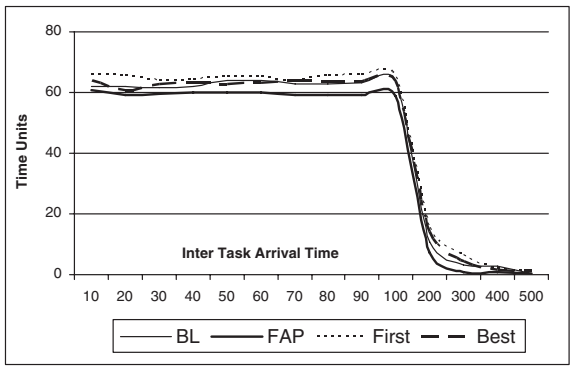
These tasks were queued and placed in arrival order to a simulated FPGA of size 64×64. The time needed to load a task was determined by the availability of an empty place on the chip and the time used to configure the cells needed by the task. The configuration delay per cell was thus also a parameter and chosen to be fixed at 1/1000 $timeunit$. Note that the computation times of the suggested methods are calculated on a host computer and is not added here. Also the changes in the value of the configuration time can be considered in future work

First, the benefit of using the new fragmentation aware placement algorithm was compared with the performance of the traditional placement strategies Bottom-Left (BL), First Fit (First), and Best Fit (Best) with the same data sets without using the real time deadline time test. Then, we examine the miss ratio in a real time environment.

In Figures 5 and 6, we compare different performance



(a) Waiting time



(b) Allocation time

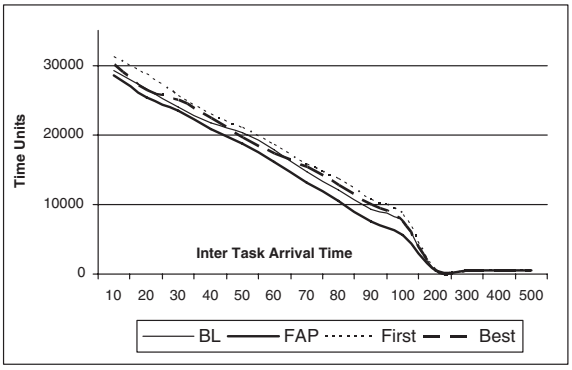
Figure 5. Performance measures (1).

measures for the traditional and the new fragmentation aware placement methodologies. An input set of data with task side up to 32 cells, and service time of maximum 500 $timeunits$ is used. The effect of varying the inter task arrival time is examined.

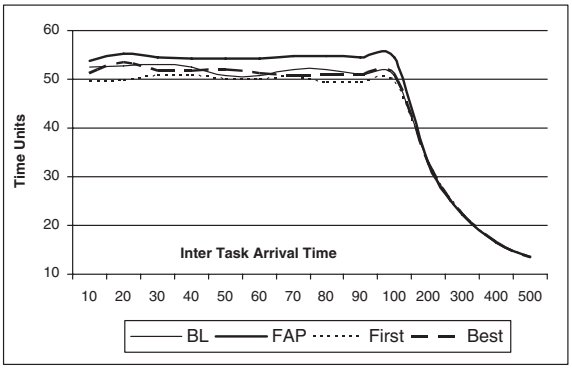
Figure 5(a) shows an improvement in waiting time of 10% over the bottom left method, 17% over the first fit method and 12% over the best fit method. In Figure 5(b), the new placement strategy has an average improvement in allocation time of 5% over the bottom left method, 9% over the first fit method and 6% over the best fit method. For the response time, Figure 6(a), we can notice that the new placement strategy has an average improvement in response time of 10% over the bottom left method, 16% over the first fit method and 12% over the best fit method. For the utilization, Figure 6(b), we can notice that the new placement strategy has an improvement up to 5% over the bottom left method, up to 9% over the first fit method and up to 6% over the best fit method.

Table 1 shows the exact measurements for different performance measures when inter task arrival time = 50 and 100 *timeunits* respectively. All measurements for waiting, allocation, and response time are in *timeunits*.

In real time environment, Table 2 shows the improvement percentage in miss ratio when using the fragmentation



(a) Response time



(b) Chip Utilization

Figure 6. Performance measures(2).

aware placement compared with the bottom left placement methodology. Task sizes are varied from 1×1 to 32×32 in the first column, from 8×8 to 32×32 in the second column, and so on. The negative values means that the FAP is worst than BL. We notice that in the first column the best improvement achieved with inter task arrival time equals 90 *timeunits*. As we increase the minimum task side length, we get more improvement in miss ratio. This is because the small tasks can be placed in some fragments in the chip and so the effect of the FAP is small. When the minimum task side length is 16 cells, the effect of the FAP is higher. The best improvement of 7.4% occurs with minimum task side of 24 cells and inter task arrival time of 200 *timeunits*.

7 Concluding Remarks

In this paper, we presented a multi-dimensional fragmentation metric. We used this metric in 2-dimensional structures to improve area utilization of FPGAs. Fragmentation aware placement methodology is introduced and its performance is compared to some standard placement strategy: Bottom Left, First Fit, and Best Fit. Our placement methodology produces an improvement between 5% and 10% over other placement strategies in chip utilization. An improvement of 16% in response time is achieved.

The effect of fragmentation aware placement in real time systems is tested. An improvement of up to 7% is achieved in miss ratio when using fragmentation aware placement.

Using this fragmentation metric in an acceptance test of the incoming tasks in real time systems is an important issue to be considered. Also, applying this strategy to heterogeneous reconfigurable systems is under consideration.

Table 1. Performance measures for different placement methodologies.

Inter-task arrival time =50 time units				
	FAP	BL	First	Best
W. time	18085	19742	20416	19005
A. time	60	64	65	62
R. time	18648	20310	20984	19571
Utilization	54.25%	50.74%	50.03%	51.93%
Inter-task arrival time =100 time units				
	FAP	BL	First	Best
W. time	5112	6950	7959	7034
A. time	58	64	65	63
R. time	5674	7517	8528	7601
Utilization	54.53%	50.73%	49.56%	50.88%

Table 2. Miss ratio improvement for FAP over BL methodology with varying task side length.

Inter-task arrival time in time units	$1 \sim 32$	$8 \sim 32$	$16 \sim 32$	$24 \sim 32$
10	-0.8%	-0.2%	1.4%	1.2%
20	-0.8%	-0.3%	1.4%	1.4%
30	0.4%	0.7%	1.2%	1.0%
40	0.9%	1.9%	1.3%	2.6%
50	-0.2%	0.5%	2.1%	1.2%
60	2.0%	2.4%	-0.1%	0.8%
70	0.4%	3.3%	2.9%	2.3%
80	1.3%	2.5%	0.2%	2.0%
90	2.8%	0.7%	2.4%	3.9%
100	1.3%	0.2%	3.1%	3.4%
200	0.4%	2.8%	4.4%	7.4%
300	0.9%	1.4%	4.5%	6.9%
400	0.6%	1.6%	4.0%	6.6%
500	0.0%	0.5%	4.5%	6.7%

References

[1] O. Diessel, H. ElGindy, M. Middendorf, H. Schmeck, and B. Schmidt, "Dynamic scheduling of tasks on partially reconfigurable FPGAs," *In IEE Proceedings on Computers and Digital Techniques*, volume 147, pages 181-188, May 2000.

[2] A. Ejnioui and R. F. DeMara, "Area Reclamation Metrics for SRAM-based Reconfigurable Device," *International Con-*

ference on Engineering of Reconfigurable Systems and Algorithms , 2005, Las Vegas, Nevada, U.S.A, June 27 - 30, 2005.

[3] M. G. Gericota, G. R. Alves, M. L. Silva, and J. M. Ferreira, "Run-Time Management of Logic Resources on Reconfigurable Systems," *Design Automation and Test in Europe*, pp. 974-979, March 2003.

[4] M. Handa and R. Vemuri, "Area Fragmentation in Reconfigurable Operating Systems," *Engineering of Reconfigurable Systems and Algorithms, Las Vegas, NV*, June 2004, Department of ECECS, University of Cincinnati.

[5] M. Handa and R. Vemuri, "An Efficient Algorithm for Finding Empty Space for Online FPGA Placement," *Design Automation Conference*, June 2004, San Diego, CA, pp. 960-965.

[6] Renqiu Huang, Ranga Vemuri, "On-Line Synthesis for Partially Reconfigurable FPGAs," *18th International Conference on VLSI Design held jointly with 4th International Conference on Embedded Systems Design (VLSID'05)*, 2005, Kolkata, India pp. 663-668.

[7] Manuel G. Gericota, Gustavo R. Alves, Miguel L. Silva, Jos M. Ferreira, "On-line Defragmentation for Run-Time Partially Reconfigurable FPGAs," *FPL 2002*, LNCS 2438, pp. 302-311, 2002.

[8] H. Walder and M. Platzner, "Non-preemptive Multitasking on FPGAs: Task Placement and Footprint Transform," *The 2nd International Conference on Engineering of Reconfigurable Systems and Architectures*, June 2002, pp. 24-30.1998.

[9] G. Wigley and D. Kearney, "The Management of Applications for Reconfigurable Computing Using an Operating System," *conference on Computer systems architecture*, 2002, In Proceedings, volume 6, pages 73-81, 2002.