International Journal of Computer Applications (0975 – 8887)
Volume 94 – No.8, May 2014

Performance Analysis of Various Fragmentation Techniques in Runtime Partially Reconfigurable FPGA

Senoj Joseph
Assistant Professor,
Dept of Electronics and Communication engineering
Sri Krishna College of Technology, Coimbatore-641042, India

K.Baskaran, PhD
Associate Professor,
Department of Computer Science Engineering,
Government College of Technology, Coimbatore, India

ABSTRACT

Reconfigurable devices, such as Field Programmable Gate Arrays (FPGAs), are very popular in today’s embedded systems design due to their low-cost, high-performance and flexibility. Partially Runtime-Reconfigurable (PRTR) FPGAs allow hardware tasks to be placed and removed dynamically at runtime. A novel 2D area fragmentation metric that takes into account feasibility of placement of future task arrivals is presented. Simulation experiments indicate that proposed technique yield better results than existing fragmentation estimation techniques when used in fragmentation aware placement.

General Terms

Algorithms, VLSI

Keywords

Fragmentation, Partially Reconfigurable FPGA

1. INTRODUCTION

Current reconfigurable devices have the ability to reconfigure parts of their hardware resources without interrupting normal operation of the remaining fabric. The placement algorithms need to find locations for placing arrival tasks and to maximize the utilization of the resources. Careless placement of incoming tasks causes portions of chip area to be wasted because they are too small to hold another incoming task. Consequently, area fragmentation is one of the biggest obstacles of obtaining good utilization of chip resources. In this paper, a new metric for measuring area fragmentation is proposed. 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 fragmentation.

Section 2 discuss about fragmentation of resources. Section 3 presents a survey of fragmentation estimation techniques Section 4 describes the new fragmentation metric Section 5 introduces evaluation of fragmentation on a sample placement output and results of comparison. Section 6 makes some concluding remarks.

2. FRAGMENTATION OF RESOURCES

In this section fragmentation is defined. In software domain this problem occurs in memory allotment technique for which efficient techniques have been developed. Partially reconfigurable FPGAs allow multitasking which leads to fragmentation of FPGA resources. Empty area on the FPGA can be covered by a set of overlapping empty rectangles. To place a task of fixed dimension the placement algorithm should identify an empty rectangle sufficiently large to accommodate the task. If such space is not available then the task will be rejected or scheduled later depending on the placement algorithm. Due to dynamic addition and deletion of tasks the empty area on the FPGA will be split into large number of small sized regions which cannot accommodate even an average sized task. This phenomenon is called fragmentation.

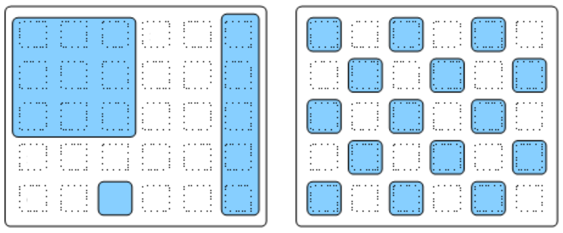

Figure 1. Two fragmentation states having same number of empty slots

Consider the following example shown in Fig 1. Both figures have the same number of empty slots but distribution varies. Classical techniques assume that fragmentation exist only when a task get rejected. Accordingly if an incoming task of height and width 3 and 2 units will be rejected in both cases. Therefore both case will be assigned with same fragmentation index but visually it shows that figure on the right is having a very high fragmentation such that it can only accommodate task of size 1 x 1. The proposed technique tries to rectify this anomaly by considering the feasibility of placing tasks of different size and assume that fragmentation exist even when task is placed.

2.1 Types of Fragmentation

There are four type of fragmentation

Internal fragmentation: This is the empty area created inside the boundary of rectangular task by trying to fit the actual logic of the task into a rectangular area.

External fragmentation: This is the fragmentation of empty area outside the boundary of the rectangular tasks. This has been already explained in previous section.

Virtual fragmentation is a situation in which the placement algorithm fails to locate contiguous empty space even though such a space exists. This can be solved by improving the efficiency of placement algorithm.

Partition fragmentation: this will occur only if FPGA surface is divided into finite number of predefined sub-area called blocks and only a single task can be placed in any of these blocks. Empty area occurs if the task placed in smaller than the block size. It is very difficult to find an optimal size of block. In literature several works have been attempted with fixed and variable sized blocks, splitting and merging of blocks etc.

The proposed work deals only with external fragmentation and the term fragmentation is used to refer to it.

3. SURVEY OF FRAGMENTATION TECHNIQUES

Fragmentation is the major reason for poor placement quality of the online placement algorithms. In this section several techniques proposed in literature are considered to estimate fragmentation and thereby improving their placement algorithms. Some metrics mix occupation degree with fragmentation. An FPGA with a high occupation but with all free area concentrated as a single rectangle cannot be considered as fragmented as some algorithm classifies.

Wigley et. al [4] defined the shortest side of an maximal empty rectangle as characteristic dimension. Fragmentation is calculated by taking the mean of distribution of characteristic dimensions. This is having less discrimination because it gives same value for several fragmentation situations.

Walder et. al. [5] found all non overlapped rectangles required to cover the empty area. Fragmentation $F = 1 - \frac{\sqrt{\sum_i a_i^2}}{\sum_i a_i}$ where i is suffix where a_i is the area of the ith non-overlapping rectangle. This may lead to different fragmentation value for the same situation based on the section of non-overlapping rectangle.

Julius Gehr, Jorg Schneider [7] suggested I D fragmentation: This is useful if the FPGA is modeled as ID array in which all the tasks have same height and variable width.

$F = 1 - \frac{1}{n}$ where n is the number of free blocks

$F = 1 - \frac{\max(f)}{\sum_{i=1}^{n} f_i}$ use max free block size in Numerator

$F = 1 - \frac{f_{avg}}{\sum_{i=1}^{n} f_i}$ use average size of free block. Mathematically identical to first one since $f_{avg} = \frac{1}{n}\sum_{i=1}^{n} f_i$

$F = 1 - \frac{\sum_{i=1}^{n} f_i^{\,p}}{(\sum_{i=1}^{n} f_i)^p}$ p=1, 2...n

A. Ejnioui and R. F. Demara [2] proposed a metric for fragmentation. Let the FPGA chip have N x N cells. Assume that a hole i consist of k cells. Therefore $f_i = \frac{k}{N^2}$. Overall fragmentation is $= 1 - \left(\prod_{i=1}^{j} f_i\right)$. Using this method the smallest possible fragmentation will be an empty chip which consists of single hole having N2 cells.$F = 1 - f_i = 1 - \frac{N^2}{N^2} = 0$. Highest possible fragmentation resembles a checkerboard pattern. If N is even there will be $\frac{N^2}{2}$ holes where each hole occupies a single cell. Therefore$F = 1 - \frac{1}{(N)^{N^2}}$. F approaches one as N gets larger. In this case also two different fragmentation states may give the same fragmentation factor.

Handa and Venuri [8] proposed a method in which they calculate FCCx and FCCy for each cell in MER.

TFCC= FCCx + FCCy.

$$\text{FCCx} = \begin{cases} 1 - \frac{v_x}{L_x - 1} & \text{if } v_x \le L_x \\ 0 & \text{otherwise} \end{cases}$$

$$\text{FCCy} = \begin{cases} 1 - \frac{v_y}{L_y - 1} & \text{if } v_y \le L_y \\ 0 & \text{otherwise} \end{cases}$$

where Lx(Ly) are average width of the tasks being placed and $v_x(v_y)$ is the number of consecutive empty cells in the horizontal and vertical direction of the current cell. Total fragmentation is the average value of TFCC of all cells in the MER. Place task in any of the corners of MER having largest TF and corner chosen to maximize the TF of task sized rectangle. Nothing is mentioned about the overall fragmentation index.

Tabero [9] proposed the Vertex list method. They measure fragmentation of each hole of empty cells after task placement which may have more than four corners, instead for each MER. $F = 1 - \prod_i \left[\left(\frac{4}{V_i}\right)\left(\frac{A_i}{A_f}\right)\right]$where V_i is the number of vertices of hole, A_i is the hole's area size and A_f is the total size of the free area. $(4/V_i)$ represent suitability of hole H_i to accommodate rectangular tasks. Any hole with 4 vertices will have best suitability. (A_i/A_f) represent the relative hole normalized area. This method penalizes holes with irregular shapes and small sizes.

Septien et. al. [6] proposed a perimeter quadrature approach. Assume that ideal hole should have a perfect square shape. Estimate how far its shape is near perfect square. Divide area A by the area of a perfect square having same perimeter p.

$A_q = \left(\frac{p}{4}\right)^2$ Relative quadrature $Q = \frac{A}{A_q}$ Fragmentation F=1-Q.Smaller one will have less fragmentation. In case of multiple holes $P = \sum P_i$and $A = \sum A_i$ and calculate as above.

J.Cui et al [3] proposed a metric based on MER that take into account the probability distribution of width and height of future arrivals instead of average values in [8]. Also they calculate the time averaged area fragmentation and use look ahead technique to choose a location with min TAAF among all candidates at the current time and next few points when some tasks complete.

ElFarag et al [1] proposed a new fragmentation model for 1 and 2 dimensional FPGA. For 8x8 array the max fragmentation will be 64 and min will be 2.

One dimensional $F = \frac{1}{m_1} + \frac{1}{m_2} + \cdots$where m_i is size of each vacant slots

Two dimension F=Fr + Fc $F_r = \frac{1}{m_1} + \frac{1}{m_2} + ..$

$$F_c = \frac{1}{n_1} + \frac{1}{n_2} + ..$$

Applying 1D method for rows and columns

4. PROPOSED METHOD

To check the efficiency of the fragmentation a fragmentation matrix is created. Consider all the MER. The row and column are labeled from 1 to n where n is the size of the FPGA. A cell (i, j) =1 indicates that a task of width i and height j can be placed on the FPGA. Example if (2, 4) =1 then a task of width 2 and height 4 can be placed on the FPGA. The pseudocode for filling the fragmentation matrix is shown below

```
Initialize x to null matrix
Find all the MER in the FPGA
For each MER of size (wi,hi)
Put x(i,j)=1 if $i<=w_i$ and $j <= h_i$
Find the number of ones in fragmentation matrix denoted by C.
Divide this by total number of empty cell. This ratio is denoted by Q.
Fragmentation F=1-Q
```

This matrix method gives the fragmentation of all possibilities.ie for same value of empty cells the fragmentation state with max possibilities (higher C) will be judged as having less fragmentation. The proposed method gives absolute fragmentation metrics. It takes 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. The above algorithm can be run in relative metric mode by changing size of X equal to (nh, nw) where nh and nw are average height and width of incoming tasks. 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.

Consider a 16 x 16 FPGA having 128 empty cells. Based on the location and clustering of empty cell the fragmentation changes as shown in Table 1. In first two case all the empty space are contiguous leading to very low fragmentation while checkerboard case results in max fragmentation.

Table 1. Some test scenario to check fragmentation

S.No	Empty rectangles in each test case(width, height)	Fragmentation
1	[16 8]	0
2	[8 8; 8 8]	0.5
3	[8 8; 8 8;16 1]	0.4375
4	[8 8; 8 8;16 3]	0.3125
5	[8 8; 8 8;16 5]	0.1875
6	[8 8; 8 8;16 7]	0.0625
7	[8 8; 8 4;8 2;4 2 ;4 2]	0.5
8	Checkerboard pattern	0.9922
9	Stripped pattern [16 1]	0.875
10	[5 4;8 6;3 3;7 4;3 1;10 2]	0.5938
11	[14 4; 5 6; 8 3;9 2]	0.4844
12	[16 4; 4 16; 4 4]	0.125

5. EXPERIMENTAL RESULTS

Experiments are performed to verify whether the fragmentation metric is consistent with other types. The simulation is carried out for 100 tasks. Fragmentation is calculated for the instances shown in Figure 3. Table 2 gives the fragmentation calculated for various test cases. The graph plotted for various test cases is shown in Figure 2. The FPGA size is set to 16x16. The fragmentation is calculated by other methods by converting the representation into a format which is suitable for their calculation. The fragmentation figure is normalized to make comparison easy. The results show that the results are consistent with other methods and give better performance when included in fragmentation aware placement. ElFarag shows very low fragmentation for all the cases while some others show very high fragmentation for test cases 9-12. Proposed method shows moderate fragmentation for such cases because tasks get rejected in not due to fragmentation but due to lack of empty space. For worst case like checkerboard pattern proposed fragmentation approaches unity.

6. CONCLUSION

An efficient model for estimation of fragmentation in online placement scenario is presented. This model can be used to benchmark other fragmentation techniques. The proposed method performs better than other techniques. The proposed technique considers only the feasibility of placing future tasks. Other features can be included to improve the performance of the metric.

7. ACKNOWLEDGEMENT

Authors thank the SKCT management and acknowledge the immense help received from the scholars whose articles are cited and included in references of this manuscript. The authors are also grateful to authors / editors / publishers of all those articles, journals and books from where the literature for this article has been reviewed and discussed.

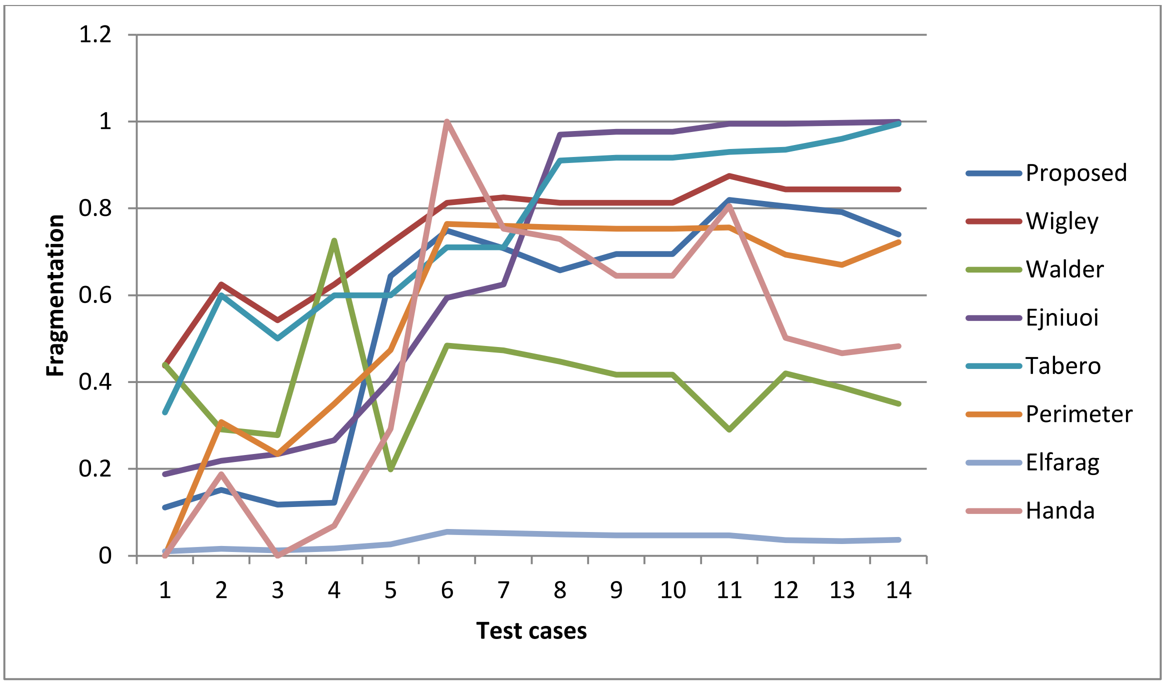

Figure 2. Fragmentation calculated for various Test cases shown in figure 3

Table 2 Simulation results for fragmentation

Test cases	**Proposed method**	**Wigley**	**Walder**	**Ejnioui**	**Tabero**	**Perimeter**	**Elfarag**	**handa**
1	0.1109	0.4375	0.44	0.188	0.33	0	0.01	0
2	0.152	0.625	0.291	0.219	0.60	0.308	0.016	3.6
3	0.1175	0.542	0.2775	0.234	0.50	0.234	0.012	0
4	0.1222	0.625	0.726	0.266	0.60	0.35	0.017	1.32
5	0.644	0.72	0.199	0.406	0.60	0.474	0.026	5.61
6	0.7485	0.8125	0.484	0.594	0.71	0.764	0.055	19.18
7	0.7083	0.825	0.473	0.625	0.71	0.76	0.052	14.45
8	0.6570	0.8125	0.447	0.97	0.91	0.756	0.049	13.99
9	0.6950	0.8125	0.4169	0.976	0.917	0.753	0.047	12.36
10	0.6950	0.8125	0.4169	0.976	0.917	0.753	0.047	12.36
11	0.8194	0.875	0.29	0.995	0.93	0.756	0.047	15.44
12	0.8047	0.844	0.42	0.995	0.935	0.693	0.036	9.63
13	0.7917	0.844	0.3877	0.997	0.96	0.67	0.034	8.95
14	0.74	0.844	0.35	0.999	0.995	0.722	0.037	9.26

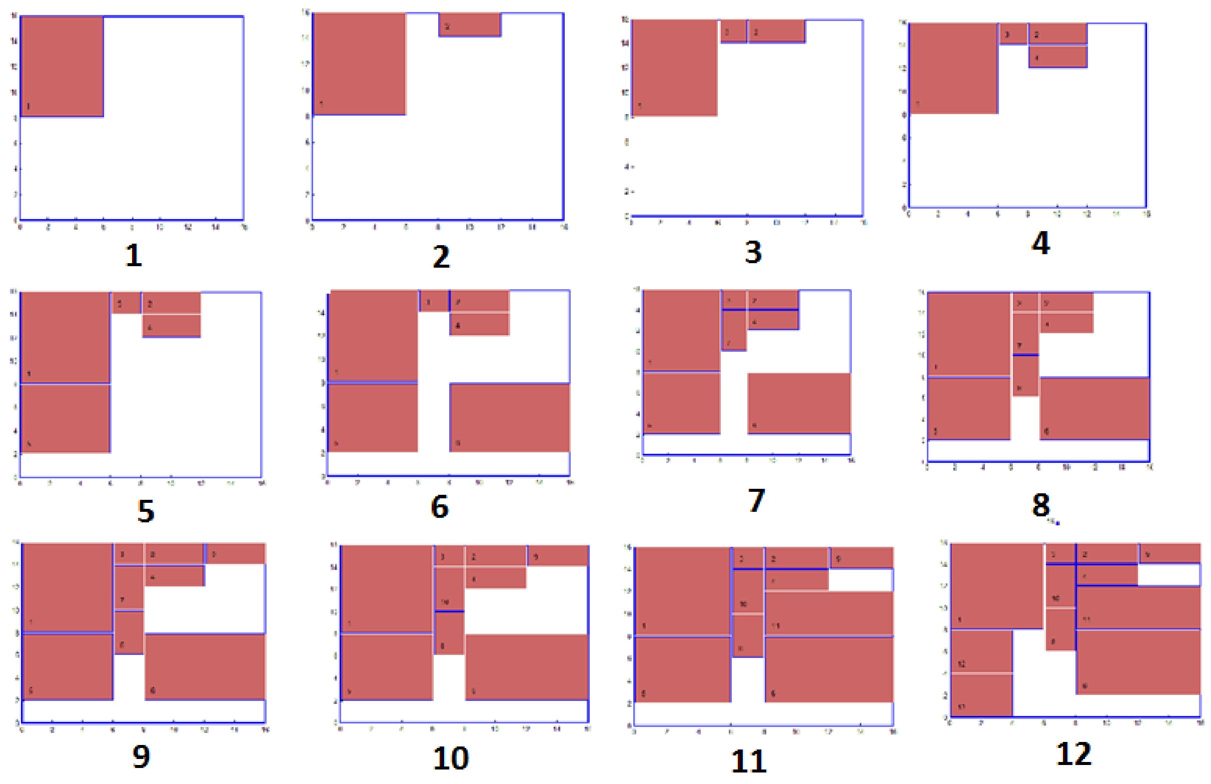

Figure 3. Various test cases considered

8. REFERENCES

[1] A. ElFarag, H. M. El-Boghdadi,and S. I. Shaheen, "Fragmentation aware placement in reconfigurable devices," in Proceedings of the 6th International Workshop on System on Chip for Real Time Applications, Dec. 2006, pp. 37-44

[2] A. Ejnioui and R. F. DeMara, "Area Reclamation Metrics for SRAM-based Reconfigurable Device," in Proceedings of the International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA'05), Las Vegas, Nevada, U.S.A, June 27 – 30, 2005.

[3] Cui, J., Gu, Z., Liu, W., & Deng, Q. (2007). An efficient algorithm for online soft real-time task placement on reconfigurable hardware devices10th international symposium on object and component oriented real time distributed computing, 2007.

[4] G. Wigley and D. Kearney. The Management of Applications for Reconfigurable Computing Using an Operating System. In Proceedings of the seventh Asia-Pacific conference on Computer systems architecture, volume 6, pages 73–81, 2002

[5] Herbert Walder and Marco Platzner. Non-preemptive Multitasking on FPGA: Task Placement and Footprint Transform. In Proceedings of the 2nd International Conference on Engineering of Reconfigurable Systems and Architectures (ERSA), pages 24–30. CSREA Press, June 2002.

[6] J. Septién, D. Mozos, H. Mecha, J. Tabero and M. A. García de Dios ,Perimeter Quadrature-based Metric for Estimating FPGA Fragmentation in 2D HW Multitasking, IPDPS 2008

[7] Julius Gehr, Jorg Schneider, "Measuring Fragmentation of Two-Dimensional Resources Applied to Advance Reservation Grid Scheduling" 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, p 276-283

[8] Handa. M , R. Vemuri, "Area Fragmentation in Reconfigurable Operating Systems", Proceedings of International Conference on Engineering of Reconfigurable Systems and Algorithms, pages 77-83, CSREA Press, 2004.

[9] Tabero, J., J.Septian, H.Mecha and D.Mozos, 2004.Low fragmentation heuristics for task placement in 2D RTR hardware management. The 14th International Conference on field programmable logic and application, FPL 2004, pages 241-250, Belgium Sept 2004.