International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 5 Issue: 3 19 – 25

Efficient Algorithms for Online Task Placement on Runtime Partially Reconfigurable FPGA

Dr.Senoj Joseph
Department of ECE
Sri Krishna College of Technology
Coimbatore
senojjoseph@skct.edu.in

***Abstract*—**Recent generations of FPGAs allow run-time partial reconfiguration. One of the challenging problems in such a multitasking systems is online placement of task. Many online task placement algorithms designed for such partially reconfigurable systems have been proposed to provide efficient and fast task placement. In this paper two different approaches are being used to place the incoming tasks. The first method is uses a run-length based representation that defines the vacant slots on the FPGA. This compact representation allows the algorithm to locate a vacant area suitable to accommodate the incoming task quickly. In the proposed FPGA model, the CLBs are numbered according to Peano Space filling curve model. The second approach is based on harmonic packing. Simulation experiments indicate that proposed techniques result in low ratio of task rejection compared to existing techniques.

Keywords*-Partial Reconfiguration; Task Placement; Free Space Management; Peano Curve; FPGA; Bin packing*

*****

I. INTRODUCTION

Reconfigurable devices with partial reconfiguration capabilities allow multitasking applications on a single chip. Embedded applications like cryptography, video communication, image processing etc. can exploit this capability. Efficient placement and scheduling algorithm can improve FPGA resources utilization and overall execution time of applications.

One of the most interesting problems is to decide where to locate the bitmap of a new task in the FPGA when it must be run. A data structure is required to keep track of available free area and the algorithm must find out the best location for the arriving task, trying to use the reconfigurable area as efficiently as possible. In online placement system, due to dynamic addition and deletion of tasks, the empty area of FPGA becomes highly fragmented and FPGA area cannot be utilized efficiently.

In this study a new data structure based on one dimension run-length encoding is developed to manage the empty area. Using this data structure placement algorithm can locate suitable location to place the incoming task quickly. A new fragmentation metric gives an indication of continuity of free space. The FPGA surface is modeled by a matrix coded according to Hilbert curve. The results show significant improvement over placement using well known algorithms like Bottom left, 2D adjacency based placement, Least interference fit technique and C Look algorithm.

This study is organized as follows: Section 2 presents an overview of problem of Scheduling and Placement in Dynamic Reconfigurable devices. A brief review of various placement and scheduling techniques are given in Section 3. In section 4, a new technique called Peano curve based placement is proposed. Section 5 describes about the bin packing techniques for online placement algorithm. Section 6 describes about the experimental setup made for performance analysis. Results and discussion are presented in section 7, followed by conclusions.

II. SYSTEM MODEL

The proposed online placement system model is as shown in Figure 1 which consists of Host CPU and partially reconfigurable FPGA. The reconfigurable resources in FPGA are a set of CLB organized in a two dimensional array. The placement module running on the host CPU consists of scheduler, placer and loader. The scheduler determines which of the tasks in the module library should be loaded and executed next. The placer will manage free space and find out optimum placement for the task. The loader loads the configuration data of tasks in the FPGA. When a task completes the resources occupied by it will be released.

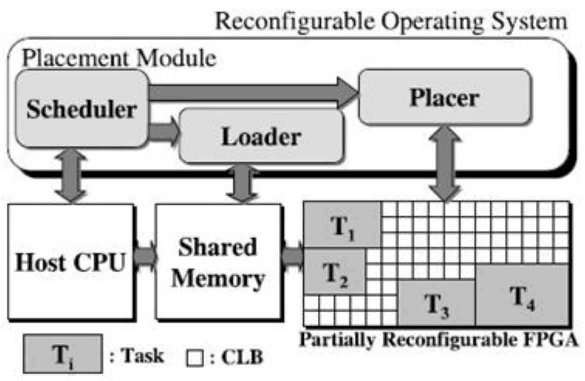


Figure 1 System model

The system assumes that the tasks arrive online, queued and placed in arrival order. As long as free area is available in the FPGA the incoming task will be placed on an unoccupied area on the FPGA. It there is no free space and the task cannot be delayed then the task is rejected. A good placement algorithm should reduce rejection rate.

The tasks are non-pre-emptive. Once a task is loaded onto the FPGA, it runs to termination. The tasks should be independent without any precedence constraints. These task parameters are defined as: for a task ti = (hi, wi, ai, si, di, xi, yi), hi and wi represent its height and width respectively and are measured in number of cells, ai, si and di are the task arrival time, execution time and deadline time. The rectangular area assigned to the task by its top left corner (xi, yi) where xi: row number and yi: column number. The size, arrival time, execution time and deadline are uniformly distributed in a predefined region and a-priori unknown.

III. LITERATURE SURVEY

An algorithm for managing free space by keeping track of non-overlapping rectangles is proposed in Ahamadenia et al [1]. The main disadvantage is that the number of empty rectangles produced quickly increases with more task insertions. This can lead to some tasks being rejected even though there is enough space to accommodate them but this space is divided between two non-overlapping rectangles. To solve this problem, they presented the idea of allowing overlapping of the empty rectangles, specifically overlapping maximal empty rectangles MERs. For n tasks, we can have O (n) non-overlapping rectangles and in the case of MERs we can have O (n2) rectangles.

Walder et al [2] proposed three partition algorithms based on Bazargan method: Enhanced Bazargan, on the fly and enhanced on the fly. The third is based on a 2D hashing table to find a feasible task placement with a run time complexity of O(1), but they did not account for reconfiguration time and also they did not account for the update time needed to update the hashing table.

Ahamadenia et al [3] proposed Horizontal line algorithm in which two horizontal lines are used: one above and another below the placed tasks. They also presented a free space management based on contour of union of rectangles algorithm. Staircase algorithm was suggested by Handa and Vemuri [4] for finding the maximal empty rectangles. Bottleneck is time for constructing staircase and finding MERs. Vertex lists was used to store free space [5] where each vertex is a possible location for an input task. Module connectivity to the remainder of the system is taken into account in [6]. Scan line algorithm was proposed by [7]. But finding maximum key elements and MER is time consuming. An intelligent merging technique to speed up Bazargan algorithm without losing its placement quality was proposed in [8]. It is a combination of three techniques selected based on the task characteristics. The techniques are: Merge only if needed, partial merging and direct combine. Deng et al [9] proposed an algorithm which packs tasks densely called 2D and 3D adjacency method. A CLook and CSAF method was proposed in [10]. Senoj and Baskaran [11-13] proposed space filling curve method for online placement task.

IV. PROPOSED PEANO CURVE TECHNIQUE

Sophisticated mapping functions have been proposed in the literature. One, based on interleaving bits from the coordinates, which is called z-ordering was proposed. By interleaving bits we get another curve called Gray code curve. A third method, based on the Hilbert curve has been proposed in literature. All these curves are having a granularity of 2 and will work for a square area which has both sides even. In this paper we show a curve which works on an odd sized area. Figure 2 shows the steps in Peano space filling curve. In this method the FPGA area has been labelled in Peano curve order. A novel data structure called run-length matrix has been introduced in [11-13] to describe the target area. In the Figure 2 the shaded area indicates task already placed. The free area can be described using run-length matrix as shown below: RL ={(6,8),(16,2),(30,18),(50,2),(56,16)}.

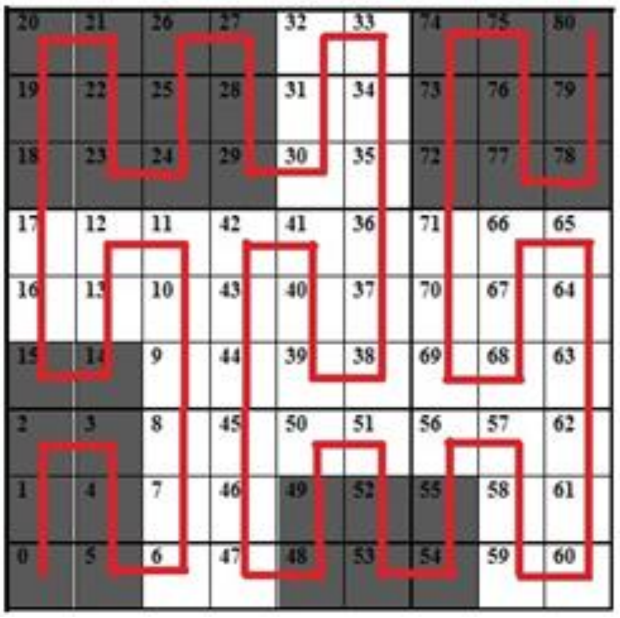
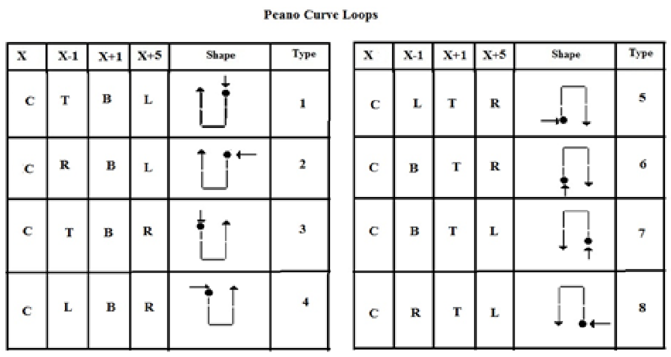


Figure 2:Peano curve with some tasks placed

Since the Peano curve is a 3 regular curve, the size of FPGA should be a multiple of 3. Each block consists of 9 cells, which are continuous. Working is similar to the other curves, but differs in the method used to identify the loops inside the curve. The Figure 3 shows the various loops identified in a Peano curve. Identifying loop using mask is similar to the Hilbert technique, but the decision is based on cells X-1, X+1 and X+5. The sub function has been adjusted to match the Peano curve. This type of curves can be suitable for odd sized tasks also. In algorithms based on area matrix methods whenever a new task is added or deleted the cells have to be recalculated. This takes considerable amount of time. The run-length will be smaller in size (worst case will be one eight of the number of CLB's) and hence less number of entries only need to be checked. Updating the run-length is also having less complexity.


Peano Curve Loops

X	X-1	X+1	X+5	Shape	Type
C	T	B	L		1
C	R	B	L		2
C	T	B	R		3
C	L	B	R		4
C	L	T	R		5
C	B	T	R		6
C	B	T	L		7
C	R	T	L		8

Figure 3: Loops occurring in Peano curve

The quality of placement algorithm can be improved by finding all feasible solutions and then selecting one based on fragmentation. Best fit find the fragmentation index for all the feasible solutions and place the task in a position that reduces the resulting fragmentation. First fit method tries to place task in the first available location that can accommodate the incoming task. It doesn't guarantee optimal result because it is a heuristic and the future inputs are unpredictable.

V. PROPOSED BIN PACKING BASED TECHNIQUES

A bin packing problem requires packing of a set of objects into a finite number of bins of capacity V in a way that minimize the number of bins used. A bin is empty if no items are packed into it else it is used. Bin packing is an NP hard problem. For a general bin packing problem, it is assumed that number of items and their sizes are known before the packing begins. A common situation is that the items come in some order and must be assigned to some bin as soon as they arrive without any knowledge of the remaining items. This situation comes under the category called online bin packing problem. Online bin packing is difficult owing to the fact that unpredictable item sizes may appear. Therefore the performance of the online bin packing algorithm is substantially affected by the permutation of items in a given list. The online task placement problem on a partially reconfigurable belongs to this category with a few changes. The number of bins is kept constant and the fact that items will be removed from the bin when they complete execution, creating vacant slot in their place which can be utilised for the incoming tasks. The FPGA area has to be partitioned into bins. The widths of bins are not constant, but decided based on the application problem.

Several algorithms exist in literature for bin packing. Probably the simplest among them is NextFit algorithm which looks only at the most recently open bin. If the item fits it is packed to that bin else it will be put in a new bin and the other is never considered again. The algorithm FirstFit scans the bins in the opening order and put the item in the first bin with sufficient capacity. If no bin is found, then a new bin is opened to accommodate the item. BestFit works similarly, but puts the item in the open bin with the least remaining capacity that is sufficient for the item. The tool used to compare online bin packing algorithm is competitive analysis. Competitive analysis compares the performance of a given online algorithm with that of an optimal offline algorithm. The competitive ratio of NextFit is 2 and that of BestFit and FirstFit is 1.7. The Harmonic algorithm by Lee & Lee [14] classifies item by size and put different items into the same bin if and only if they belong to the same class. Harmonic algorithm and it extensions achieve the best known competitive ratio of 1.589 [15].

Consider an FPGA of size 32x32. The inputs are task with their area specified. The FPGA is divided into bins of equal width, e.g., 2 CLB. Let the maximum size of the bin be 32 CLB. The task of height 'i' will be assigned to a bin 'i'. Hence this algorithm is called Fixed height (FH). If it cannot be accommodated, then the task will get rejected. The maximum size of task is taken as 32 CLB so that bin can hold at least two large sized tasks. The advantage is that searching for empty locations can be restricted to only one bin, and if there exist a vacant space it will surely fit, because in that bin all the tasks kept will have the same height. The drawback with this method is that some of the bins will be heavily used as shown in Fig 4.a.

The second method named Harmonic Height (HH) is based on harmonic algorithm. The bins are having constant width. The total number of bin is obtained by dividing FPGA width by bin width. In this case also the maximum size of the incoming task is assumed to be 32 CLBs. The height of each incoming task will be calculated by dividing the area of task with the bin width. There are four types of bins based on height of the tasks that can be included in that bin. This classification is done according to the harmonic algorithm. Type 1 bin will accommodate tasks with height less than 3, type 2 bin will have task with height [3 to 4], type 3 bin will accommodate tasks with height [5 to 7] and type 4 bin will accommodate tasks with height [8 to 16]. These four types of bin will be repeated 'N' times shown in (1) where W is width of FPGA, T is number of bin types and B is the width of each bin. This method has better performance than the FH as shown in Figure 4.b.

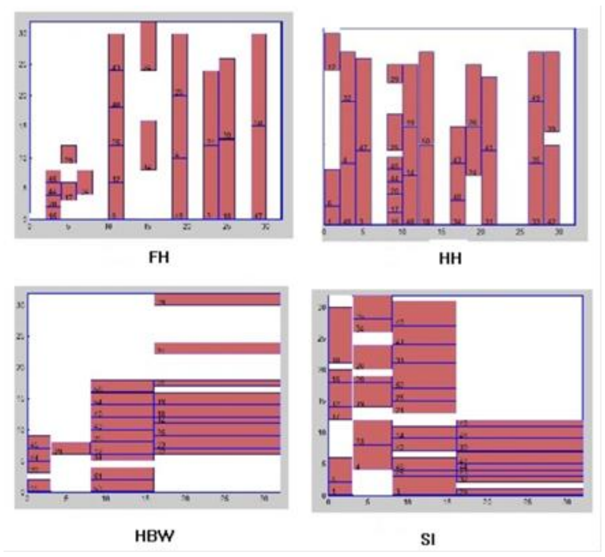


Figure 4: (clockwise a,b,c,d) : Screenshot for Bin packing based placement

$$N = \frac{W}{T * B} \qquad (1)$$

The third method named as Harmonic Bin Width technique (HBW) is having bins with different width made according to harmonic algorithm. A FPGA of width 32 will be divided into bins of width 3, 5, 8, 16. If the task size is greater than 16 put it in the largest bin. If size is between 8-15 use the bin of size 8. For task size 5-7 use bin of size 5 and smaller tasks use bin 3. The simulation is performed using a test bench having 300 tasks. The intertask arrival time is in range [0-50] execution time range is [0-300], slack [0- 50] etc. The simulation is repeated with different test set to study the effect of inter-task arrival time, slack and task size. It is found that the algorithm leads to internal fragmentation. The screenshot is shown in Figure 4c.

The HBW method suffers from internal fragmentation. i.e., when a task of size 17 arrives, it is assigned to a bin of width 16. Therefore the height of task will be fixed to 2 taking up an area of 32 CLBs on the FPGA instead of its actual size of 17 CLBs. A solution to this problem is to assign the task to a bin that reduces the internal fragmentation. The internal fragmentation resulting while placing tasks of size up to 32 in bins of different width was estimated. The incoming task will be assigned to a bin, which produces the least internal fragmentation. However new problems arise because it is found that the bins of width 3 and 5 get maximum number of tasks and will be filled quickly. Therefore, a question arises on how to maintain each bin equally.

This led to the idea of preparing a look up table, which gives the order of the bin choices for various task sizes. This method is named as suitability index method. For a particular task size, if the bin that comes first in the list does not having vacant space, then the next choice from the lookup table will be attempted. If all the choices have exhausted, then the task has to wait till vacant space is created in any of these bins or get rejected when it cannot meet the deadline. The screenshot is given in Figure 4d.

The task is defined by their area. The aspect ratio of the task will be decided by the algorithm by considering three important parameters. They are vacant space inside bin, internal fragmentation if the task placed in that bin, Aspect ratio (G) of the task if placed in that bin. G is ratio of height to width of task. The first condition gives more preference to bins having less internal fragmentation (I). For penalizing tasks having elongated shape, a parameter F is defined based on the aspect ratio as shown in (2). The minimum condition is used in (2) to avoid domination of F in (3). Otherwise the higher width bins will get less chance for placing tasks. The third parameter makes sure that all the bins are uniformly utilized. Combining all, a parameter called suitability index (S) is defined as (3), where K is percentage of height occupied in that bin and M is the maximum possible value for internal fragmentation. M will be calculated as largest bin width minus 1.

$$F = \min(1, |1 - g| \qquad (2)$$

$$S = \left(\frac{I}{M}\right) + F + K \qquad (3)$$

To illustrate this method, let us assume an FPGA of size 32x32, let the occupancy level of bin3, bin5, bin 8 and bin16 be 0.312, 0.1875, 0.125 and 0.0625, respectively. Maximum possible fragmentation is 15, which is for a bin of width 16. Let the task to be placed have an area 30. For bins with width 3 and 5 internal fragmentation will be zero. If placed in bin8 and bin16 the internal fragmentation is 2 units and 2 units, respectively. For bin of width 3 the aspect ratio is 3.3 since the height is 10. For bin of width 5 the aspect ratio is 1.2 since height is 6. The aspect ratio value indicates that the best choice for the task is bin 5. If all the parameters have equal weightage the suitability index is calculated as given below, which shows that it is better to put the task in bin5 even though the bin3 have the least internal fragmentation and bin16 has lowest occupancy.

Bin3 S=0+1+0.3125=1.3125
Bin5 S=0+0.2+0.1875=0.3875
Bin8 S= (2/15)+0.5+0.125= 0.758
Bin16 S= (2/15) +0.875+0.0625=1.07.

VI. EXPERIMENTAL SETUP

Simulation framework has been done using Matlab 7.8 running on 2.2 GHz Intel core i3 processor. The simulation is done using randomly generated data for evaluating the algorithm. This has been done in the past because it is impossible to generate real data for future technological advancement. In this section we present two methods: the first one is a fast placement and another fragmentation aware placement technique. These techniques are compared with standard placement techniques like Bottom left, 2D adjacency based placement, Least interference fit technique, EAC [16] and Clook algorithm. Bottom left is a classical bin packing algorithm which places the incoming task first empty slot available starting from bottom left corner of FPGA. 2D adjacency based technique choose the location for the incoming tasks to make tasks placed ”densely”, in order to have larger continuous free area remains. The 2D-Adjacency of a Candidate Cell equals to the number of adjoining tasks/boundaries of the incoming task if the Base Cell of the incoming task is placed here. Least interference technique will select a location which minimize the number of columns disturbed to minimize the number of running tasks getting halted while reconfiguration. Clook method is explained in Lee et al. [10].

In order to evaluate the effectiveness of algorithm simulation is performed for an FPGA with 27x27 CLB. This model is adopted because the previous studies most relevant to this work used FPGA of similar size for their simulations and the space filling curve works on surface with size power of two. Sixty sets of 500 tasks each are randomly generated for each experimental environment and the results shown in next section are the average over these sets. The height and width of the tasks are chosen randomly between 1 and a maximum value of 9 CLBs. Lifetime of the tasks is generated randomly between 1 and 500 time units. Delay between two consecutive tasks is also chose between 1 and user defined L time units. The workload can be controlled using different upper bound L. A smaller L means that the tasks arrivals are more frequent and FPGA area utilization is higher. All parameters are assigned by sampling a uniform random distribution function in their respective validity intervals

The following assumptions are used in this work. The tasks are independent and pre-emptive. Pre-emptive task is one if started cannot be stopped before its expiry. Due to this relocation of tasks is also not permitted. Since the tasks are independent they can be scheduled in any order. Rotation of task is not used. For bin packing methods the incoming task area will be given. The width and height of the task will be decided by the algorithm. The following parameters are measured to test the effectiveness of the proposed algorithm. Suppose during the simulation interval (0, T), N tasks arrived and n tasks were rejected. A task may be rejected placement if sufficient contiguous area is not available currently and it if cannot meet its deadline if scheduled at a later time. Average task rejection ratio is the ratio of number of tasks rejected to total number of tasks.

VII. RESULTS AND DISCUSSION

The simulation for the Peano curve is done on an FPGA of size 27x27. The granularity (size of the smallest task) of the task is taken as 3x3block. The size of tasks should be a multiple of three. If it is not so, the algorithm will round the size to nearest multiple of 3 thereby allotting more space on the FPGA than required. The results are similar to the other curves and a screenshot for the placement is shown in Figure 5. The colored boxes correspond to tasks that are currently running. Task that have completed is not shown. The white region indicates empty region which is already got fragmented due to placement and removal of tasks. The results are tabulated in Table 1.

The results show that the performance of the proposed placement matches with conventional method for all cases. The rejection rate was more for large sized task as expected. The rejection rate increases with decrease in inter-task arrival time range. When tasks arrive in quick succession then more number of tasks will be running on the FPGA leaving less room for the newly arrived task. When deadline is tight then more tasks get rejected. If deadline is loose then tasks can wait as late as possible and get placed whenever a free slot is available. When slack becomes very large then none of the tasks get rejected. The simulations of all the algorithms are run with 25 task sets each having 200 tasks and the average value of each parameter is shown in Table 2. The task sets are generated with different values for execution time range, slack and inter-task arrival time period. HH and SI are found to be having better performance in all the parameters.

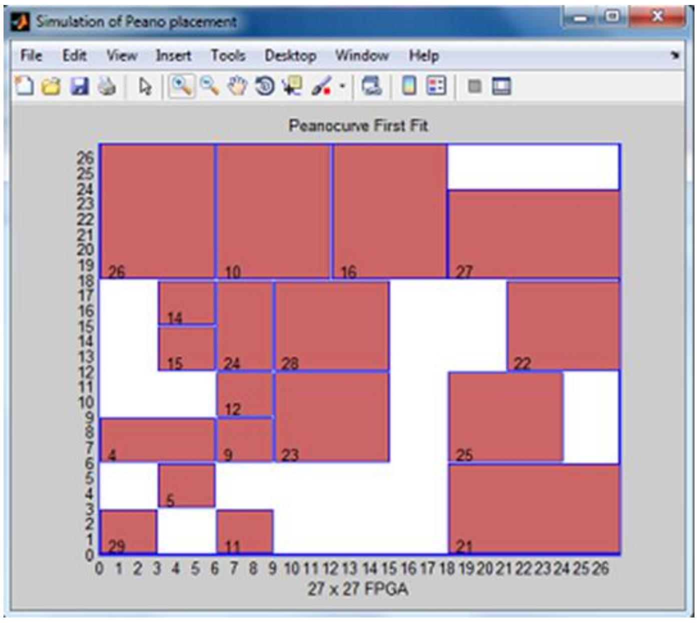

Simulation of Peano placement
File Edit View Insert Tools Desktop Window Help
Peanocurve First Fit
26
10
16
27
14
15
24
28
22
12
4
9
23
25
5
29
11
21
27 x 27 FPGA

Figure 5: Screenshot of Peano curve

VIII. CONCLUSION AND FUTURE ENHANCEMENT

In this study a new approach for scheduling and placement of task on a dynamic reconfigurable device based on Peano space filling curve method is being presented with the goal of minimizing task rejection ratio and increasing FPGA utilization. A second approach using bin packing technique is also proposed. Both methods work well compared with existing techniques in terms of rejection ratio. They are both scalable to work on bigger area matrices.

ACKNOWLEDGEMENT

The Author thanks the SKCT management and acknowledges the immense help received from the scholars whose articles are cited and included in references of this manuscript. The author is 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

REFERENCES

[1] A. Ahmadinia, C. Bobda, M. Bednara and J. Teich, A new approach for on-line placement on reconfigurable devices, Proceedings of the 18th International Parallel and Distributed Processing Symposium, IEEE Xplore Press, pp: 134-140, April. 26-30, 2004

[2] Walder, H., C. Steiger and M. Platzner, Fast online task placement on FPGAs: Free space partitioning and 2D-hashing, Proceedings of the IEEE International Parallel and Distributed Processing Symposium, IEEE Xplore Press, pp: 178-178, Apr. 22-26, 200.

[3] A. Ahmadinia, C. Bobda, S.P. Fekete, J. Teich and J.V.G.Veen, Optimal free space management and routing conscious dynamic placement for reconfigurable devices, IEEE Trans. Comput., 56: pp. 673-680, 2007

[4] M. Handa, and R. Vemuri, An efficient algorithm for finding empty space for online FPGA placement, Proceedings of the 41st annual Design Automation Conference, San Diego, pp: 960-965, Jun. 07-11, 2004

[5] J. Tabero, J. Septian, H. Mecha and D. Mozos, A Low Fragmentation Heuristic for Task Placement in 2D RTR HW Management, Proceedings of the 14th International Conference field programmable Logic Application, IEEE Xplore Press, Belgium, pp: 241-250, Belgium, Aug. 30-Sep. 1, 2004.

[6] M. Tomono, M. Nakanishi, S. Yamashita, N. Nakajima and K. Watanabe, A new approach to online FPGA placement, Proceedings of the 40th Annual Conference Information Sciences Systems, IEEE Xplore Press, Princeton, pp: 145-150, Mar. 22-24, 2006.

[7] J. Cui, Q. Deng, X.Q. He and Z. Gu, An efficient algorithm for online management of 2D area of partially reconfigurable FPGAs, Proceedings of the 7th IEEE International Conference Design, Automation and Test in Europe Conference Exhibition, IEEE Xplore Press, pp: 1-6, Apr. 6-20, 2007.

[8] T. Marconi, Y. Liu, K. Bertels and G. Gaydadjiev, Intelligent merging on-line task placement algorithm for partial reconfigurable system, Proceedings of the 8th Design, Automation and Test in Europe, IEEE Xplore Press, Munich, pp: 1346-1351, Mar. 10-14, 2008.

[9] Q. Deng, F. Kong, N. Guan, L. Mingsong and W. Yi, On-line placement of real-time tasks on 2D partially run-time reconfigurable FPGAs. Proceedings of the 5th IEEE International Symposium Embedded Computing, IEEE Xplore Press, Beijing, pp: 20-25, Oct. 6-8, 2008.

[10] T.Y. Lee, C.C. Hu and C.C. Tsai, Adaptive free space management of on-line placement for reconfigurable systems," Proceedings of the International Multi-Conference Engineers Computer Scientists, Hong Kong, pp: 322-326 Mar 17-19, 2010.

[11] Senoj Joseph & Baskaran, K, An FPGA Task Placement Algorithm Using Reflected Binary Gray Space Filling Curve, International Journal on Reconfigurable Computing, vol. 2014 pp.1-7

[12] Senoj Joseph & Baskaran, K, A Temperature Aware Z-Curve Based Online Task Placement Algorithm for Partially Reconfigurable FPGAs, Journal of Theoretical and applied Information technology, vol.66, no.3, pp. 861-866, 2014 ISSN 1992-8645

[13] Senoj Joseph & Baskaran, K, An online task placement algorithm using Hilbert curve for a partially reconfigurable Field programmable gate array', TENCON 2015 IEEE region 10 Conference

[14] Lee, CC & Lee, DT 1985, A simple online bin packing algorithm, Journal of the ACM, 32(3), pp. 562-572

[15] Benjamin, H & Tjark, V 2011, Probabilistic alternatives for competitive analysis, ZIB-report 11-55, Berlin, Germany

[16] Iturbe, X, Benkrid, K, Arslan, T, Hong, C & Martinez, , Empty resource compaction algorithms for real time hardware task placement on partially reconfigurable FPGAs subject to fault occurrence', Proceedings of International Conference on Reconfigurable computing and FPGAs, pp. 27-34, 2011.

Table 1:Rejection rate for various workloads

	Rejection rate (%)					
Inter-task arrival time period (time units)	**2DA**	**BL**	**CL**	**LIF**	**EAC**	**PFF (proposed)**
10	49.00	49.13	48.00	49.17	49.40	48.13
25	33.88	35.20	33.24	35.60	35.44	34.20
50	19.72	20.52	18.80	20.64	20.36	18.28
75	11.08	12.04	10.44	12.20	12.44	10.16
100	7.52	7.88	7.00	8.16	9.04	6.60
125	5.72	6.40	5.24	6.76	6.56	5.12

Table 2:Performance of Bin packing algorithms

	Program execution time	**Number of task rejected**	**Average waiting time for the tasks (Time units)**	**Total schedule time (time units)**
FH	24	90.16	41.96	1084.40
HH	27.2	6.88	58.68	1116.68
HBW	24.4	21.88	27.12	1113.48
SI	49	8.64	2.6	1073.28