

A Fast First Fit Hardware Task Placement Algorithm For Partially Reconfigurable FPGAs Using Windowing-Or-Logic Method

B. Premalatha¹ and Dr. S. Umamaheswari²

¹*Assistant Professor, Department of ECE,
Coimbatore Institute of Technology, Coimbatore, Tamil Nadu, India-641014.
Mobile:9994714595, email: premalathaece.cit@gmail.com*

²*Associate Professor, Department of ECE,
Coimbatore Institute of Technology, Coimbatore, Tamil Nadu, India-641014.
Mobile:9944756014, email: umamaheswari.cit@gmail.com*

Abstract

Exponential growth of demand in VLSI based design of digital electronic systems for real life applications can be satisfied by using reconfigurable Computing Systems such as FPGAs. Recent FPGAs allows multitasking and the task can be freely inserted and deleted for and after computation without disturbing the other tasks computation which is called partially reconfiguration. The process of inserting tasks inside FPGA for its computation is called hardware tasks placement in which algorithm execution time and task rejection ratio are two major issues yet to solve. Tasks are placed inside FPGAs based on any one of fitting strategies such as First fit, Best fit and Worst fit, introduces fragmentation inside the FPGA area which is also an issue to solve during placement. Free space availability is an important factor to consider before the task has to place inside FPGA for its computation. Various data structures are used to maintain the free space availability of an FPGA. In placement process, time taken by the free space search algorithm will be more that reduces the performance of placement algorithm. This paper focused on reducing the execution time of hardware task placement algorithm by using a new method called “Windowing OR Logic” in First fit mode. In addition other factors such as task rejection ratio, resource utilization and waiting time on an average are also measured. Proposed method provides fast and easy free space searching to place the task. Various approaches of first fit mode placement algorithms are compared with proposed method and simulation results shows the algorithm execution time of proposed one improved in the range of 1.5 -4 times for small scaled FPGA

devices of size 8x8, 16x16 and in range of 5- 10 times better for large scaled FPGA devices of size 32x32, 64x64 and 100x100.

Keywords:- PR FPGAs, Compact Hardware Task Placement, free space search algorithm, Algorithm Speed, VLSI.

I INTRODUCTION

Highlighting the alternative method of high performance computing in real life applications between the ASIC (application-specific Integrated circuit) and GPP (General Purpose processor) as Reconfigurable FPGA based computing system is proved to be good choice. Reconfigurable FPGAs has the advanced features like high degrees of parallelism, more flexibility and easy to programmable. Due to advancement in VLSI Technology, the FPGAs are available in various scaled size with computational elements as resources in both homogeneous and heterogeneous form. The application has been downloaded into FPGA through JTAG controller, and it has to place inside resources of FPGA for its execution. For proper usage of limited resources inside the FPGA, effective online scheduling and placement algorithm has to be developed. Scheduler and Placer comes under the function of overall management of its Reconfigurable Computing operating system. Scheduler schedules the task for execution, only if the placer gives the good command about the free space availability of FPGA resources. Scheduler and placer tied together decide the placement performance. To speed up the application execution, Reconfigurable Computing Operating system requires suitable placement algorithms with fast free space search methods. Time Complexity in free space search depends on the type of data structure used to represent the free space. We aimed at developing such fast free space search algorithm with new approach using Boolean OR-Logic called "Windowing OR-Logic" methods. The organization of paper is as follows: Motivation and proposed work is discussed in section 2. Related work is discussed in Section 3. About hardware task model, Placer and Time complexity is discussed in Section 4. Types of Fitting strategies and FPGA representation is discussed in Section 5. Different FPGA placement algorithms are discussed in section 6. Experimentation and results are discussed in section 7. Result analysis is shown in Section 8. Finally, the Conclusion is discussed in Section 9 followed by References.

II MOTIVATION AND PROPOSED WORK

FPGAs have limited resources for its execution and so the operating system has to manage the resource allocation to incoming application's tasks in effective and intelligent manner. More research works has been carried out in both on-line and offline strategy of scheduling and placement of tasks and their performance is proven by simulation results. As per the literature survey, online FPGA placement algorithm satisfies the following constraints (i) algorithm should results less task rejection ratio (ii) less waiting time (iii) faster execution time (iv) High resource utilization. Real time tasks sets with non-preemption and rectangular in shape are randomly generated

for experimentation in this paper. Speed of execution plays major role to decide the placement quality. Previous works related to Scheduling and placement algorithm provides the solution for the following issues . (i) First, How to manage the fragmented free space for task placement as because dynamic insertion and deletion of tasks introduce fragmentation inside the FPGA area? Fragmented free space management by MER based method is time consuming one. (ii) Second, using various fragmentation metrics, the fragmentation aware placement is introduced to obtain better quality placement under best fit strategy. This approach also time consuming, it takes time to find all possible positions followed by calculating the low fragmentation metric to improve the placement quality. (iii) Third, different placement methodologies are found such as stuffing, Region based, Quad corner etc, All the above categories motivates how to devise an algorithm for free space search within FPGA area. Always first fit based placement outperforms in terms of speed. From the literature Survey, it motivates us to find the fast and easy free space search algorithm to increase the speed of application execution. So, a new approach in First fit mode called "Windowing OR-Logic" is introduced and proved that it requires less time for application execution. No need to maintain MER in proposed method. Experiment was carried out in simulation tool, and the comparison is made between the basic existing placement approaches such as bottom left, C-look in modified way, MER based First fit, MER based Best fit placement methods. In addition, new various approaches are found for the first fit strategy based placement and compared. "Windowing OR-Logic" method outperforms in terms of speed especially for large scaled FPGAs which are most widely used in Industries.

III RELATED WORK

Kirash Bazargan et. al introduced fast online placement methods for dynamic reconfigurable systems using MER based and the performance is discussed in terms of speed and task acceptance rate [1]. This paper is the basis for all the researchers work in this field. Herbert Walder and Marco Platzner, discussed the basic placement techniques such as first fit, best fit and worst fit and they introduced the footprint transform method for FPGA Placement. Execution time performance is compared for all placement techniques[2]. Christoph Steiger et al. , proposed guarantee based scheduling for two situations like tasks with arbitrary and synchronous arrival time. List of free rectangles are managed by new method called Bazargan's partitioning [3]. Herbert Walder and Marco Platzner, FPGA area is partitioned into blocks of various sizes in dynamic way, and the scheduling methods such as SRPT(Shortest remaining Process time) and EDF(Earliest Deadline First) are used. Block partitioning based heuristic is found to improve the placement quality[4]. A. Ahmadiania & J. Teich proposed the method that gives the solution for currently available FPGAs limitation like reconfigurable overhead. Method named Least Interference Fit (LIF) is introduced and it places the tasks at the location where the less number of tasks are running already [5]. Christoph Steiger et. al. , introduced two heuristics such as horizon and stuffing as online scheduling method suitable for both 1D and 2D FPGAs and the performance results are shown [6]. M. ToMono et. al. , proposed an method

including both degree of fragmentation and speed of I/O Communication based on Manhattan Distance [7]. A. Ahmadiania et. al, proposed Nearest Possible Position (NPP) method for space management [8]. It maintains the occupied space rather than free space in FPGA area. This method gives the solution where the number of empty rectangle increased than the placed rectangles. Ali Ahmadiania et. al, proposed cluster based placement algorithm in which the clusters are found according to mobility interval and they compare the performance with KAMER based Placement algorithm[9]. M. Handa and R. Vemuri proposed the Staircase Algorithm for finding MERs. Every CLB in 2D array which is FPGA area contains the positive number that gives the number of continuous empty cells in the column above including the cell itself [10]. Jin Cui et. al. , presents a novel online task placement algorithm for minimizing fragmentation on FPGAs and they introduced one level look ahead heuristic for high quality placement[11]. Jin Cui, Qing Xu Deng, Xiuqiang He, Zonghua Gu, proposed a new algorithm for finding the complete set of rectangles using Scan line algorithm with different encoding technique to represent FPGA. and its performance are compared with respect to execution time[12]. A. A. Elfrang et. al, discussed about the best fit placement by fragmentation aware metric. Metric used is multi dimensional one and is used to improve the area utilization[13]. Comparison is made between placement strategies such as Bottom left, First Fit and Best fit and its outperformance is proved. X. Zhou et, al, introduced new method called Compact Reservation, which outperforms the 2D stuffing method[14]. Thomas Marconi etal. , proposed an intelligent stuffing based placement method to improve the response time and reduce the wasted area. Performance results are dictated as 31. 3 % improvement in response time, 1. 5% shorter schedule time and 89. 7 % total wasted area gets reduced[15]. Thomas Marconi, YiLu, Koen Bertels, Georgi Gaydadjiev, discussed about the best way to search the place for new tasks in right place and right time within short time and they found algorithm called intelligent merging(IM) and the performance results are shown[16]. Jesus Tabero et. al. , vertex list set for free space boundary is found. Fragmentation and adjacency are the two heuristics used to find the best location. Look a head heuristics is presented for placement[17]. J. Septien etal. , introduced a new technique to estimate the free area fragmentation based on quadrature of free area perimeter and no need to maintain ERs in free area as in other methods[18]. Mostafa Elbidweihy Thesis, proposed region based scheduling and placement method and concentrate on heterogeneous FPGAs schedulers design[19]. Y. Xiao, Z. Duan and P. Nie proposed ESLA algorithm for free space management [20]. Trong-Yen Lee etal. , works on new methods to find the candidate space of rectangular or non rectangular as CLook and CSAF . Performance highlights in terms of Task rejection ratio, total execution time and waiting time[21]. Chuan Hong, proposed a novel light footprint and fast execution allocator for dynamically placing hardware tasks into FGAs. Introduced new placement algorithm called EAC (Empty Area Compaction) with First fit allocator and the simulation result outperforms for different FPGA Sizes[22]. Yi Lu etal. , discussed about the immediate fit algorithm to reduce the task rejection ratio and increase the placement quality[23]. Manish Handa and Ranga Vemuri, detailed about the fragmentation inside FPGA Area and introduces new fragmentation metric[24]. Maisam Mansub Bassiri and Hadi Shahriar

Shahhoseini, introduces the new concept in FPGA placement as reusing the already configured area and the Simulation methods outperforms[25].

IV HARDWARE TASK MODEL, PLACER AND TIME COMPLEXITY

Field programmable Gate Arrays (FPGAs) consists of computing resources named Configurable Logic blocks(CLBs) arranged in matrix form and the incoming application gets downloaded into FPGA for its execution. Scheduling and Placement has strong nexus between them and overall performance of task execution depends on speed. Speed of Application execution depends on how fast the task gets placed inside FPGA for execution which in turn depends on how fast the free space gets searched inside the FPGA. Once the task gets enters into scheduler, the scheduler sends an interrupt to placer otherwise called Free space manager to give the candidate location to place for execution. Based on the methods used to find the free space, over all execution time of an application varies.

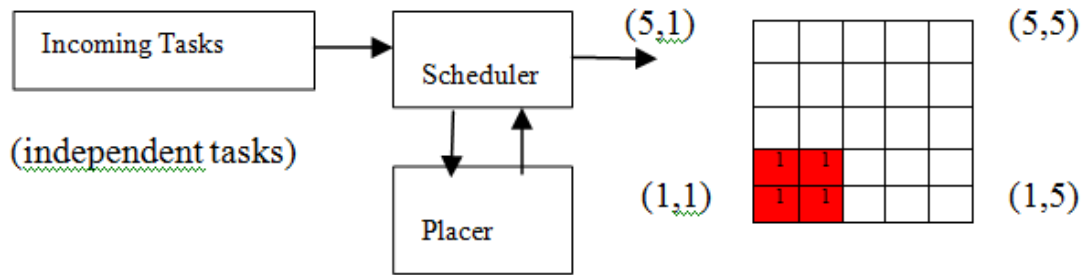


Fig. 1 Reconfigurable Computing System

IV A. HARDWARE TASK MODEL

Incoming Application consists of ‘n’ number of tasks which are independent, rectangular in shape and non-preemption(i. e. once enters the FPGA for execution, it remains till finishes its execution). Tasks are characterized by the 6-tuple parameters

$$T_i=(n_i, tar_i, tex_i, w_i, h_i, tde_i) \tag{1}$$

where n_i = i^{th} task, tar_i =arrival time of i^{th} task, tex_i =execution time of i^{th} task, w_i =width of i^{th} task, h_i =height of i^{th} task, tde_i =deadline of i^{th} task and tasks is in shape of rectangle. For each incoming tasks with arrival time tar_i and execution time tex_i its deadline to wait to get free space for execution, tde_i is found by

$$tde_i=tsch_i+tex_i+rand(1, t) \tag{2}$$

where $tsch_i$ is the scheduled time satisfying the constraints $tsch_i \geq tar_i$. if placer gives the free space co-ordinates at the time of task arrival, then $tsch_i = tar_i$ else $tsch_i > tar_i$ and t is any integer value.

IV B. PLACER AND TASK WINDOW

Placer has free space manager which includes the insertion list and deletion list updater that takes some time to search inside the FPGA for free space information for the required task width and height. Once the tasks finds place for execution, the corresponding CLBs are filled with 1's. Free space searching is time consuming and it should be fast enough to facilitate the features of reconfigurable FPGAs. List based data structures are used in Placer in most of the algorithms. Here the term 'Windowing' means the task size defined by width (w_i) and height (h_i). For e. g. , if task size=(2, 3), $w_i=2$ and $h_i=3$. The task window is shown in Fig 2.

0	0
0	0
0	0

Fig. 2 Task Window of Size (2x3)

V FITTING STRATEGIES AND FPGA REPRESENTATION

There are 3 types of Fitting strategies used in FPGA placement such as first fit, best fit and worst fit. First fit defines the very first free space rectangle that large enough to place the task. (compact) Best fit defines the smallest free space rectangle that is enough to place the task. Worst fit defines the largest free space rectangle that task can place for execution. First fit can be found by bottom left approach, scan the free space from the bottom left corner of FPGA and select the first free rectangle to accommodate the task. Best fit can also found from the bottom left approach, and select the smallest rectangle to accommodate the task. Best fit approach also depends on the fragmentation after the tasks get placed. FPGA is represented by 2D matrix, filled with 0's initially and then with 1's after task gets placed for execution.

Table 1 Fitting Strategies: BF, FF, WF for task window(2x2)

S. No	Task size	Available free space	Suitable Fitting strategy
1.	2x2	{1, 2}-4x3	FF, WF
		{3, 4}-3x3	BF, WF
		{4, 2}-2x2	BF

VI DIFFERENT FPGA PLACEMENT ALGORITHMS

Generally, the MER based method is used to find the free rectangles for placement. In this paper, Existing methods such as overlapping MER based, non overlapping MER based Placement and Least Interference fit placement methods are discussed. In addition, various methods are proposed to find the first fit candidate solution as follows (i) Bottom Left First Fit approach based on Row column Reduction method (BL-FF-RCR) (ii) Top Left First Fit approach based on Column-

Row Reduction method (TL-FF-CRC)(iii) Bottom Left –First Fit Windowing Sum-C Look (BL-FF -WS- C Look) (iv) Bottom left First Fit tree search (BL-FF-TS). (v) Bottom Left First Fit approach using Windowing OR-Logic (BL-FF-Windowing- OR Logic). For all incoming tasks($i=1$ to N) do

1. For each task arrival, find whether the space is available to place inside FPGA area is by using any one of the method. (i) list of Empty rectangles (MER Based-(OL/NOL)). (ii) any deterministic method(WS/RCR/CRC/OR Logic)
2. Based on the Fitting strategy either First fit or Best fit, the fragmentation calculation involved. If no space found at the time of arrival, task is in queue and it has the constraint that it can wait in queue for free space availability till its deadline. Once the tasks crossed its deadline it has to reject. i. e. , if $T > T_{\text{deadline}}$; task is rejected.
3. If more than one tasks are in queue, Earliest deadline First priority is used to schedule the tasks. Once the tasks finds the place and executed, it will delete from the space management list and the CLB's occupied by the tasks gets freed to accept the next tasks.
4. Occupied CLBs are represented by logic '1' / task number and Unoccupied CLBs are represented by logic '0'. Continuous task insertion and deletion in the 2D FPGA area introduces the fragmentation. Best method has to be find to manage the task placement with fragmented FPGA area. In most of existing method, MER Based free space management is used which is time consuming. To alleviate this, In this paper an Fast First fit hardware task allocation method called "Windowing-OR-Logic" is found . One-by-One, algorithms are discussed below with examples and steps involved in it.

VI A. MER BASED BEST FIT FPGA PLACEMENT (MER-BF-OL)

In this method, free space is maintained by list of overlapping (OL)Maximal empty rectangles and it is time consuming due to every time the task is inserted, the number of Empty rectangles increases. This limits the speed of Execution. In addition, to obtain the Best fit location co-ordinates, the fragmentation is used as a metric which is also an time consuming when the number of rectangles increases. As a whole, MER-BF-OL method decreases the speed of execution. In this paper, fragmentation for each rectangle is calculated by using the formulae $F = F_R + F_C$. where F_R =Row-wise Fragmentation and F_C = Column-wise Fragmentation. But F_R and F_C is found by using run length encoding method which is used to find the continuous length of ones and zeros. Rectangle with minimal fragmentation is selected for tasks placement.

Steps involved in MER-BF-OL method:

For all tasks $i=1$ to N (N =total number of tasks) do

1. After each task insertion, find the set of maximal empty rectangles (Overlapping rectangles). For Best Fit strategy based placement, calculate the fragmentation of all possible rectangles by using the Formula.

Fragmentation metric for 2D FPGA area, $F=Fr+Fc$;

Where Fr = row-wise fragmentation; Fc = column wise fragmentation.

If $Fr_1, Fr_2, Fr_3 \dots Fr_m$ etc. , are the fragmented segments in row wise, the value of Fr is calculated by using $Fr=(1/Fr_1)+(1/Fr_2)+(Fr_3)+\dots$. If $Fc_1, Fc_2, Fc_3 \dots$ etc are the fragmented segments in column wise means, the value of Fc is calculated by using $Fc= Fc_1+(1/Fc_2)+(1/Fc_3)+\dots$. Fc_n Substitute the value of F_r and F_c , to obtain the Fragmentation metric.

2. Select the rectangle which has minimum fragmentation metric. Place the tasks in the selected rectangle's co-ordinates in bottom left corner.
3. Time consumption by the algorithm execution involves (i) time to find the list of MER. (ii) time to calculate fragmentation.

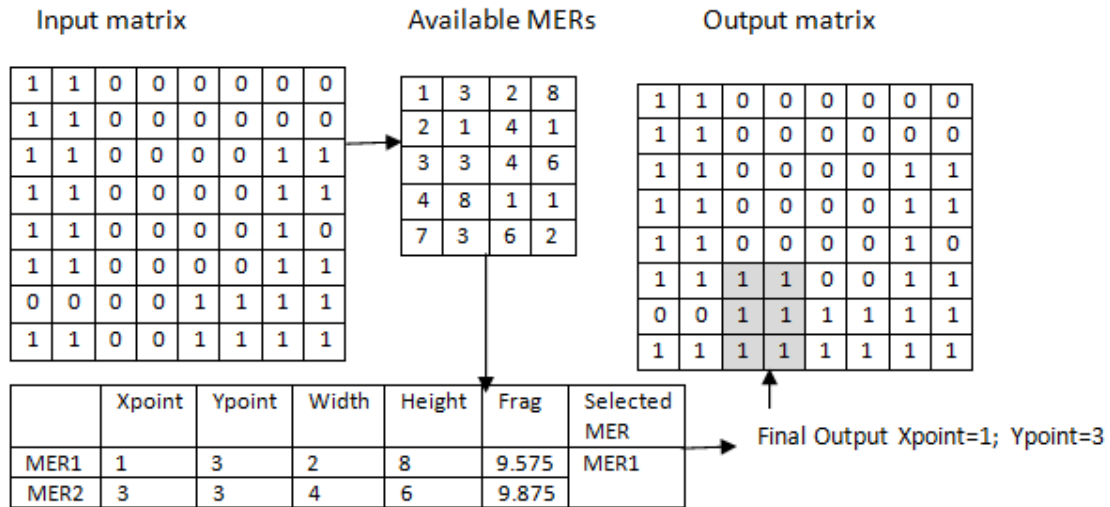


Fig. 3 Example for MER-BF-OVL Method of FPGA Placement

VI B. MER BASED BEST FIT FPGA PLACEMENT (MER-BF-NOL)

In this method, free space is maintained by list of Non-overlapping (NOL) Maximal empty rectangles and it is time consuming as every time the task is inserted, the number of Empty rectangles increases. This limits the speed of Execution. In addition, to obtain the Best fit location co-ordinates, the fragmentation is used as a metric which is also an time consuming is number of rectangles increases. As a whole, MER-BF-NOL method decreases the speed of execution. In this paper fragmentation is calculated similar to MER-BF-OL) method.

Steps involved in MER-BF-NOL method:

For all tasks $i=1$ to N (N =total number of tasks) do

1. After each task insertion, find the set of maximal empty rectangles (Non-

- Overlapping rectangles). For Best Fit strategy based placement, calculate the fragmentation of all possible rectangles by using the Formula.
2. Time consumption by the algorithm execution involves (i) time to find the list of MER. (ii) time to calculate fragmentation (iii) insertion and deletion of tasks.
 3. Select the rectangle which has minimum fragmentation metric and Place the tasks in the selected rectangle's co-ordinates in bottom left corner.

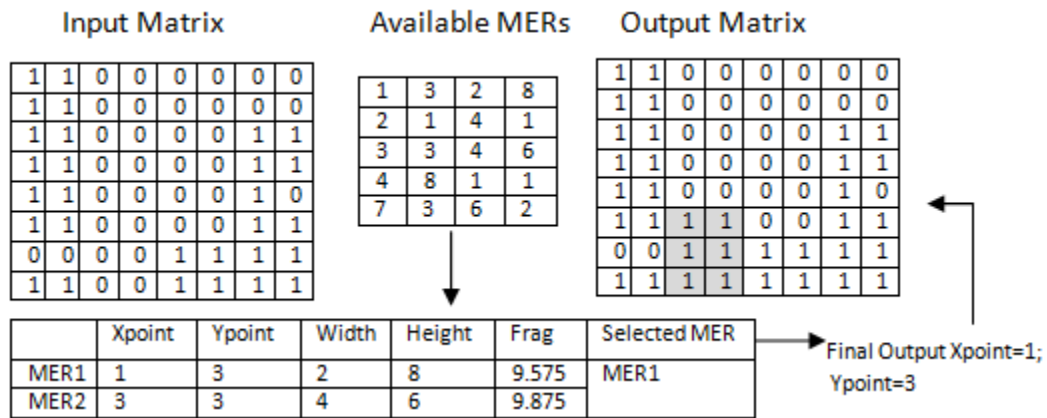


Fig. 4 Example for MER-BF-NOL Method of FPGA Placement

VI C. MER BASED FIRST FIT FPGA PLACEMENT (MER-FF-OVL)

In this method, after found the set of overlapping Maximal empty rectangles as discussed in MER-BF-OL and (MER-FF-OVL) methods, the tasks co-ordinates are found by selecting the First Rectangle from Bottom left in the list. No need of fragmentation calculation for this method.

Steps involved in MER-FF-OVL method.

For all tasks $i=1$ to N (N =total number of tasks) do

1. After each task insertion, find the set of maximal empty rectangles (Overlapping rectangles). First Fit strategy based placement is applied in this algorithm, So need to calculate the Fragmentation metric.
2. Select the rectangle which has lowest bottom left co-ordinate in the set of MER and place the tasks.
3. Time consumption by the algorithm execution involves only time to find the list of MER, insertion and deletion of tasks.

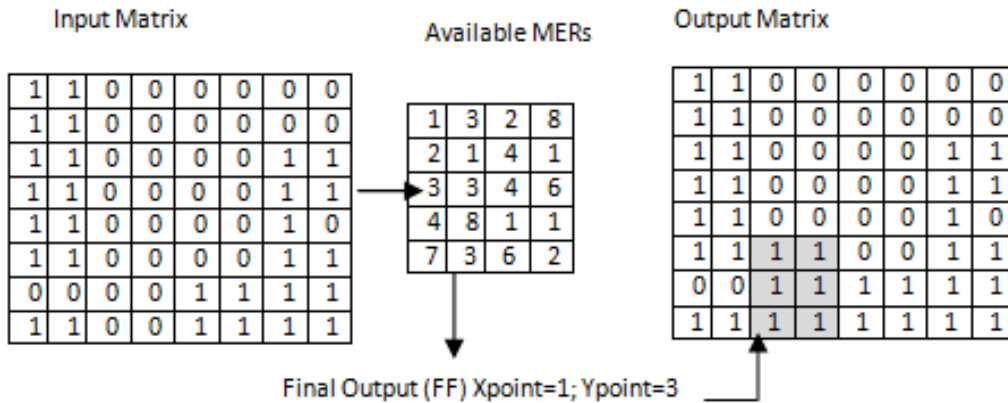


Fig. 5 Example for MER-FF-OVL Method of FPGA Placement

VI D. LEAST INTERFERENCE FIT BASED FPGA PLACEMENT (LIF)

In this method, after find the set of Empty rectangles, rectangle in which column with less number of reconfiguration required is chosen to be the one for task placement. Reconfiguration overhead can be easily solved by Least interference Fit method

Steps involved in LIF method.

For all tasks $i=1$ to N (N =total number of tasks) do

1. After each task insertion, find the set of all possible empty rectangles of size equal to required task size.
2. Find out the column in which only less number of tasks are running. Reduce the reconfiguration overhead.
3. Select the rectangle corresponding to that column selected with less reconfiguration requirement for placement.
4. Time consumption by the algorithm execution involves time to find the list of ER in all locations, time to find the least reconfigurable column, insertion and deletion of tasks. Advantage of this method is less task rejection ratio.

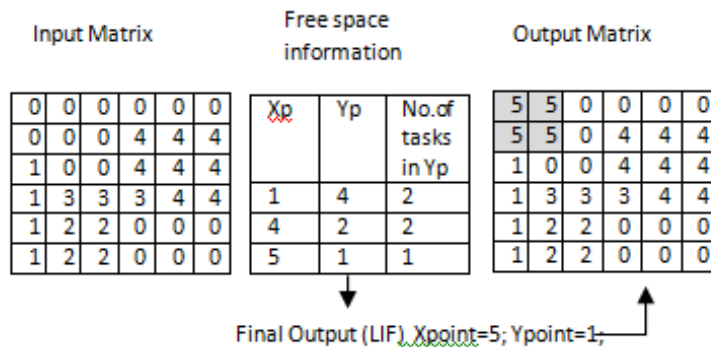


Fig. 6 Example for LIF Method of FPGA Placement

VI E. BOTTOM LEFT FIRST FIT BASED ON ROW COLUMN REDUCTION METHOD(BL-FF-RCR)

This method will be the alternative one to MER based First Fit FPGA placement methods in which the row and column of the matrix are decomposed into size equal to width and height of tasks and find whether the space is available. Scanning is done in size equal to task parameters, the time taken to find the free space will be less. But even the time complexity is more than proposed BL-FF-Windowing-OR Logic method. This method is similar to stacking the objects in shelf in horizontal manner(width based).

Steps involved in BL-FF-RCR method.

For all tasks $i=1$ to N (N =total number of tasks) do

1. In this method, no need to find the set of Empty rectangles(MER-OL/NOL)
2. Starting from the bottom row of FPGA, find the continuous free space by using modified RLE method.
3. Once the free space of required width is available, stop the row scanning. Say at R_x .
4. Find the x - location in the row at which the free space starts. If the available width is equal to required width, number of decomposition (N_d) in column wise is from that point is equal to 1. Otherwise the number of column –decomposition from the x -location is equal to $xlocation-w_i+1$;
5. For the iterations equal to N_d , find whether the space equal to tasks height is available by Row –decomposition. $N_{d1} = R_x:H-hei$.
6. Scanning procedure results the First fit strategy for tasks placement.
7. Time consumption by the algorithm execution involves the time to find the free space, row-column-row decomposition followed by insertion and deletion of tasks.

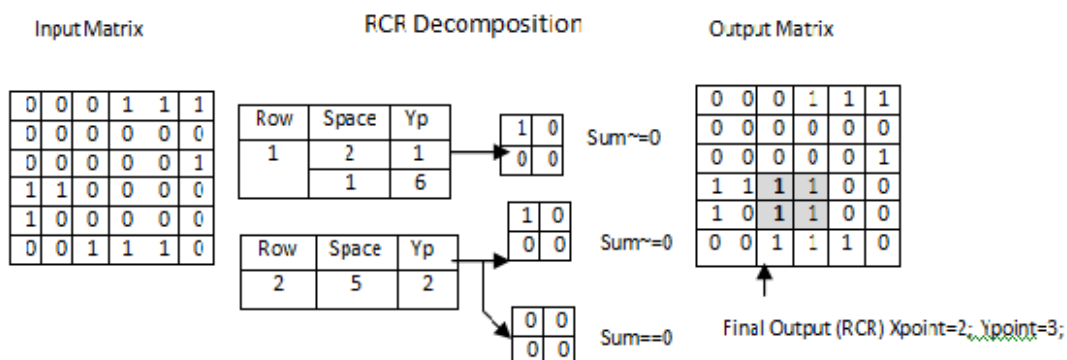


Fig. 7 Example for BL-FF-RCR Method of FPGA Placement

VI F. TOP LEFT FIRST FIT BASED ON COLUMN ROW REDUCTION METHOD (TL-FF-CRC)

This method will be similar to BL-FF-RCR method except the Column from the left side FPGA area is decomposed into size equal to task width followed by the Rows from bottom of FPGA area is decomposed into size equal to tasks height . Scanning is done in size equal to task parameters. This method is similar to stacking the objects in vertical manner. (height based) and deletion of tasks.

Steps involved in TL-FF-CRC method:

For all tasks $i=1$ to N (N =total number of tasks) do

1. In this method also, no need to find the set of Empty rectangles(MER-OL/NOL)
2. Starting from the left side column of FPGA, find the continuous free space by using modified RLE method. Once the free space of required height is available, stop the Column scanning. Say at C_x . Find the x - location in the Column at which the free space starts. If the available height is equal to required height, number of decomposition (N_d) in row wise is from that point is equal to 1. Otherwise he number of column –decomposition from the x -location is equal to $xlocation-hei+1$;. For the iterations equal to N_d , find whether the space equal to tasks width is available by Column decomposition. $N_d1:C_x:W-w_i$.
3. Scanning procedure results the First fit strategy for tasks placement.
4. Time consumption by the algorithm execution involves the time to find the free space, Coloum-row –Column decomposition followed by insertion

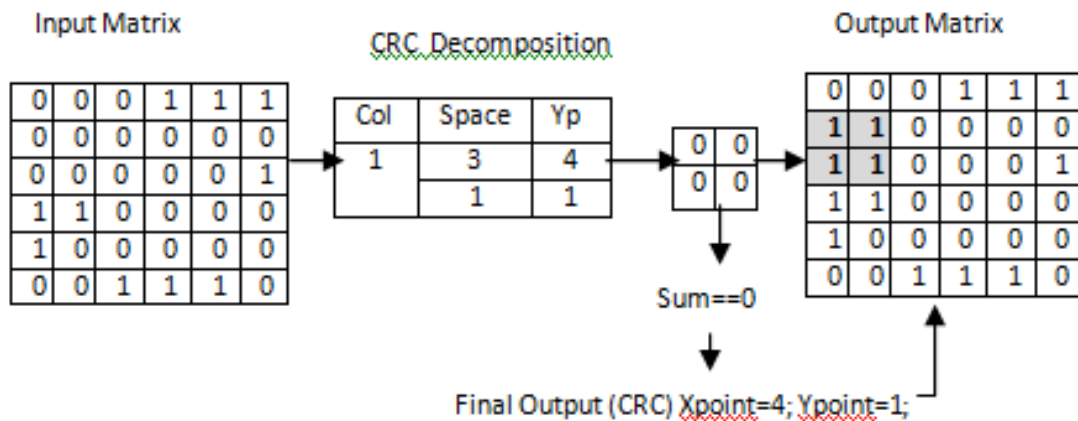


Fig. 8 Example for TL-FF-CRC Method of FPGA Placement

VI G. BOTTOM LEFT FIRST FIT USING WINDOWING SUM (BL-FF-WS-CLOOK)

In this method, instead of decomposing rows and column, from the bottom left corner the task window will be of size equal to tasks width and height then perform sum of

column wise sum called windowing sum. If the resultant value equals zero, indicates the space available of required size, otherwise proceed with next window until space found.

Steps involved in BL-FF-WS-C Look method.

For all tasks $i=1$ to N (N =total number of tasks) do

1. No need to maintain the Set of MER(OL/NOL). Starting from the bottom left corner of FPGA which is denoted as i and j (where $i=1$; $j=1$; initially), consider the window of size equal to tasks width and height, find the sum(sum(window)) called Windowing –Sum(WS). If the value of windowing sum (WS) is equal to zero, the value of i and j is taken as the candidate co-ordinates to place the tasks.
2. But if the value of WS is not equal to zero, the value of I and j gets incremented and perform WS.
3. Repeat the step 3 and 4 till the value of WS is equal to zero. Set the location to place the tasks as the value of I and j at which $WS=0$;
6. Time consumption will be equal to the number of windows from bottom left corner of FPGA area it process till the space finds.
7. Scanning range in both Rows and Column is reduced equal to $W-w_i+1$ and $H-h_i+1$ respectively, where W =width of FPGA, H =height of FPGA. Thus above steps results the First fit strategy for tasks placement.

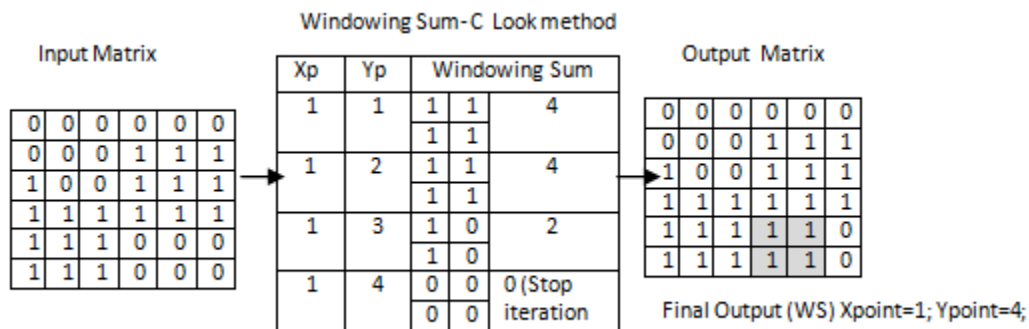


Fig. 9 Example for FF-BL-WS-C Look Method of FPGA Placement

VI H. BOTTOM LEFT FIRST FIT USING TREE SEARCH METHOD. (BL-FF-TS)

In this method, tree based free space searching algorithm is used. No need to maintain the list of free rectangles.

Steps involved in BL-FF-TS Method:

For all tasks $i=1$ to N (N =total number of tasks) do

1. Numbering the CLBs in 2D FPGA area as 1, 2, 3 etc. , from bottom left corner.

2. Find the splitted fragmented space in each row by using the modified RLE method.
3. Let the splitted space of each row can be named as S1, S2, S3 etc. , Select the first Splitted row co-ordinates (x, y), which is suitable to accommodate the tasks, then branch the scanning to the rows above and below it for prescribed height of tasks.
4. While branching, the starting point of free space in branched row will be equal to or less than target co-ordinates. Otherwise reject the row it was selected for branching to next level. 5. If row gets cancelled, proceed with next possible location from the another row, then proceed the steps.
5. This method will take time, if more new nodes arises. Anyhow, the First fit co-ordinates are found with reduced time than MER based placement methods

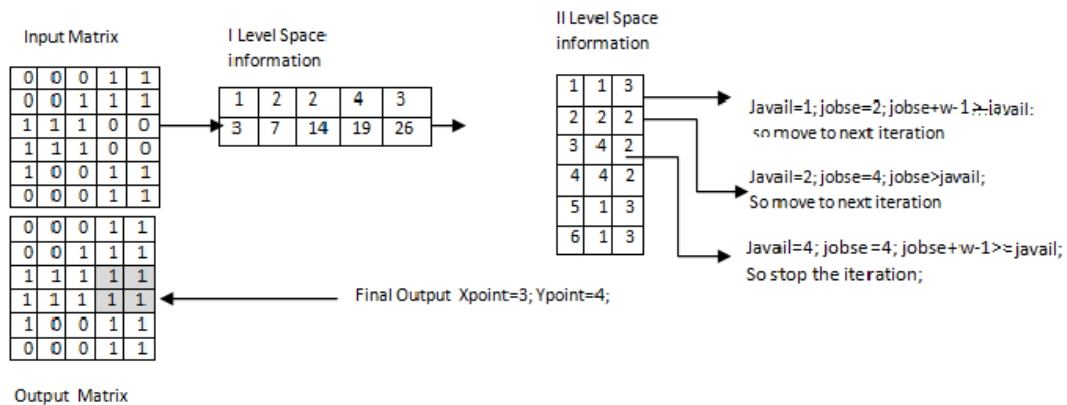


Fig. 10 Example for BL-FF-TS method of FPGA Placement

VII. BOTTOM LEFT FIRST FIT APPROACH USING WINDOWING OR-LOGIC (BL-FF-W-OR LOGIC)

Steps involved in FF-BL-OR Logic method:

For all tasks $i=1$ to N (N =total number of tasks) do

1. This method is different from FF-BL-WS method in such a way that no need to perform windowing sum from bottom left corner, instead n-input OR operation is performed for size equal to height of tasks starting from bottom Row.
2. If the result of n-input OR Operation indicates the contiguous free space (if the output of OR operation is zero) of height equal to task height and then apply string find matching concept to check whether the space of size equal to tasks width is available. (Consider string length is equal to task width).
3. From the combination of Boolean –OR Logic and String find matching concept, free space in First fit mode can be found easily and quickly.
4. Even for small size tasks, n-input parallel operation of OR Logic increases the speed of execution.

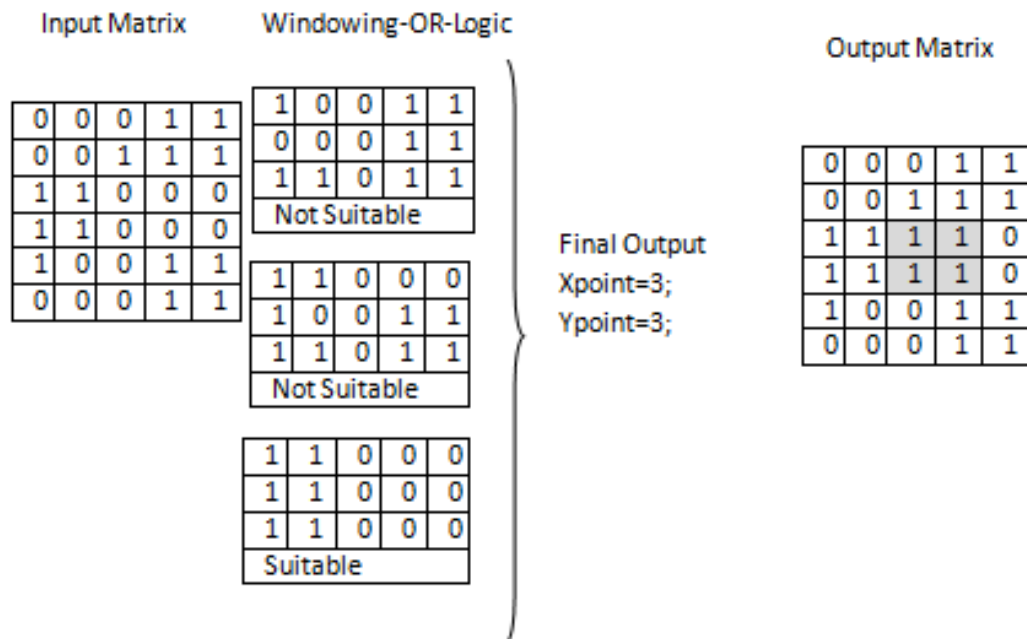


Fig. 11 Example For BL-FF-Windowing-OR Logic Method of FPGA Placement

VII EXPERIMENTAL SET-UP AND SIMULATION RESULTS

Real time environment for validating the proposed algorithm is created by generating the random tasks sets and the simulation is done in Intel® Core™ i5-3230M, 2.60 GHz Processor. All the real time tasks parameters are chosen randomly within certain ranges for FPGA size as 8x8, 16x16, 32x32, 64x64 and 100x100 and is tabulated in Table 2. Arriving time between tasks is assumed as 1 time unit. Width and height of the tasks are chosen in two categories of small load and large load. Thirty sets of each 500 tasks are generated for simulation and for Large scaled device of size 100x100, three sets of 1000 tasks are considered for simulation. Proposed algorithm called “Windowing- OR-Logic” in first fit mode outperforms the other in terms of execution time by maintain less than or equal task rejection ratio value, except for Least Interference Fit method.

*slfa= small load fast access tasks, llfa= large load fast access tasks, mlfa= medium load fast access. All the simulation results are tabulated for all FPGA sizes from Table 3 to Table 6. Simulation parameters are Task rejection ration, Total waiting time and Execution time is tabulated. Fast access means the inter-arrival time between the tasks is 1 time unit. Small access means the inter-arrival time between the tasks is more than 1 time unit. Average task rejection ratio, Average waiting time and average execution time of an algorithms are compared.

Table 2 Parameters of Input hardware tasks for experimentation

S. no	FPGA Size	tar	Wi	hei	Ex. time	Task deadline tdead
1	08x08 slfa	1-500	1-4	1-4	5-10	Tar+tex+rand(1, 5)
2	08x08 llfa	1-500	2-8	2-8	5-10	Tar+tex+rand(1, 10)
3	16x16 slfa	1-500	1-5	1-5	10-50	Tar+tex+rand(1, 10)
4	16x16 llfa	1-500	5-15	5-15	10-50	Tar+tex+rand(1, 10)
5	32x32 slfa	1-500	2-15	2-15	10-50	Tar+tex+rand(1, 10)
6	32x32 llfa	1-500	15-30	15-30	10-50	Tar+tex+rand(1, 10)
7	64x64 mlfa	1-500	5-30	5-30	20-100	Tar+tex+rand(1, 10)
8	100 x100 mlfa	1-1000	10-50	5-20	20-50	Tar+tex+rand(1, 10)

Table 3 Performance parameters for small scaled FPGAs (8x8)

<i>Algorithms</i>	8x8 slfa			8x8 llfa		
	Trr	Tw	Exec time	Trr	Tw	Exec. time
BL-FF- WOR	8. 2	8. 84	0. 6163	61. 9	5. 16	0. 7514
MER-BF-OL	7. 2	8. 94	1. 1084	61. 7	5. 19	1. 1204
MER-BF-NOL	19. 8	7. 72	1. 0067	65. 8	4. 63	0. 9243
MER-FF-OVL	7. 9	8. 87	0. 6069	61. 9	5. 15	0. 9628
BL-FF-Clook	8. 2	8. 84	0. 5761	61. 9	5. 16	0. 6540
BL-FF-RCR	8. 5	8. 81	1. 4688	62. 4	5. 09	1. 7726
TL-FF-CRC	21. 3	7. 59	0. 7097	67. 7	4. 37	1. 5066
BL-FF-TS	8. 2	8. 87	0. 4542	61. 9	5. 16	0. 7795
LIF	5. 5	39. 6	3. 3903	57. 1	5. 80	3. 8217

Table 4 Performance parameters for small scaled FPGAs (16x16)

ALGORITHMS	16x16 slfa			16x16 llfa		
	Trr	Twa	Exec time	Trr	Twa	Exec time
BL-FF- WOR	14. 9	5. 69	0. 8936	88. 3	1. 09	1. 7399
MER-BF-OL	14. 6	5. 71	4. 0707	88. 4	1. 08	3. 2295
MER-BF-NOL	31. 6	4. 57	3. 5729	89. 4	0. 99	2. 5305
MER-FF-OVL	15. 0	5. 68	1. 8033	88. 4	1. 08	3. 1195
BL-FF-Clook	14. 9	5. 69	0. 7911	88. 3	1. 09	1. 2651
BL-FF-RCR	15. 5	201	3. 5310	88. 3	123	4. 1207
TL-FF-CRC	27. 0	4. 88	1. 0367	90. 3	0. 90	4. 7529
BL-FF-TS	14. 9	5. 69	1. 1738	88. 3	1. 09	2. 6538
LIF	9. 1	6. 07	18. 276	81. 2	1. 75	29. 773

Table 5 Performance parameters for small scaled FPGAs (32x32)

Algorithms	32X32 slfa			32x32 llfa		
	Trr	Twa	Exec time	Trr	Twa	Exec time
BL-FF- WOR	40. 9	1. 28	3. 5211	92. 8	0. 24	5. 3511
MER-BF-OL	40. 7	1. 29	8. 6983	92. 8	0. 24	6. 4802
MER-BF-NOL	58. 0	0. 91	6. 1280	92. 9	0. 24	5. 4059
MER-FF-OVL	40. 8	1. 29	6. 7803	92. 8	0. 25	6. 3820
BL-FF-Clook	40. 9	1. 28	5. 7654	92. 8	0. 24	5. 0466
BL-FF-RCR	42. 0	1. 26	18. 493	92. 9	161.	10. 100
TL-FF-CRC	60. 6	2. 44	3. 6576	93. 2	0. 23	26. 583
BL-FF-TS	40. 9	1. 28	6. 1019	92. 8	0. 24	6. 9916
LIF	31. 0	1. 50	523. 34	93. 3	0. 23	342. 99

Table 6 Performance parameters for small scaled FPGAs (64x64) & (100x100)

Algorithms	64x64 slfa			100x100 mlfa		
	Trr	Twa	Exec. time	Trr	Twt	Exec. time
BL-FF- WOR	63. 1	0. 39	22. 039	28. 3	1889	43. 62
MER-BF-OL	63. 3	0. 38	31. 680	27. 8	1987	83. 5
MER-BF-NOL	74. 8	0. 26	20. 387	17. 3	1130	70. 6
MER-FF-OVL	63. 2	0. 38	29. 189	28. 2	1974	61. 54
BL-FF-Clook	63. 1	0. 39	61. 953	28. 3	1889	392. 4
BL-FF-RCR	64. 1	557.	126. 60	29. 4	1977	559. 6
TL-FF-CRC	78. 9	0. 22	16. 666	64. 9	14877	57. 39
BL-FF-TS	63. 1	0. 39	35. 028	28. 3	1889	101. 2
LIF	52. 0	0. 50	157. 46	15. 6	1739	271. 1

VIII RESULT ANALYSIS

Simulation results for various FPGA sizes of this proposed method suits well for large scaled FPGAs

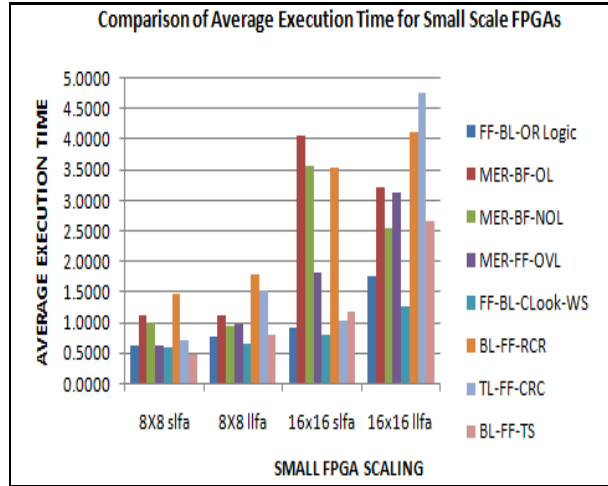


Fig. 12 Comparison of Average execution time of Proposed OR-Logic method with others for Small scale FPGAs

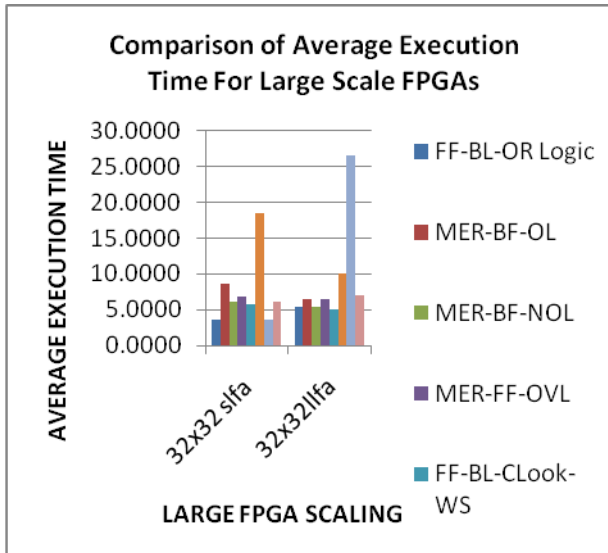


Fig. 13 Comparison of Average execution time of Proposed OR-Logic method for Large Scale FPGAs

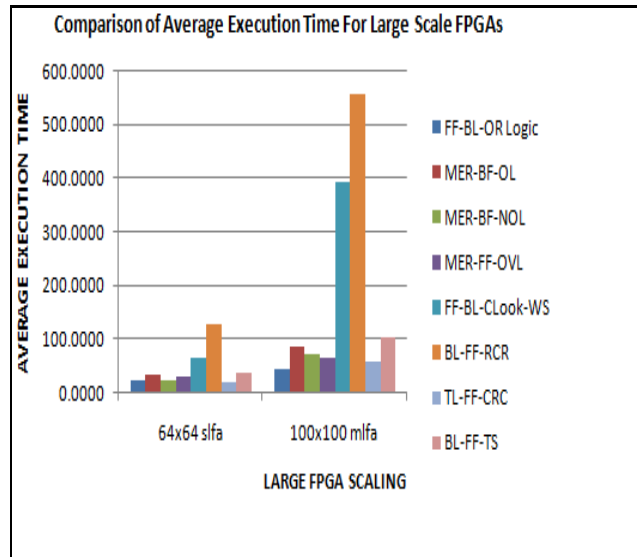


Fig. 14 Comparison of Average execution time of Proposed OR-Logic method with others for Large scale FPGAs(64x64) and (100x100)

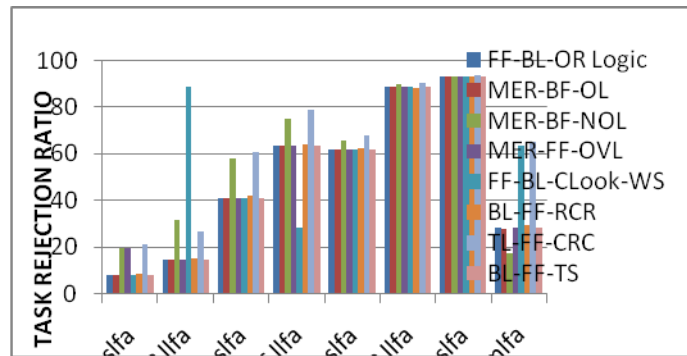


Fig. 15 Comparison of Task rejection ratio of Proposed OR-Logic method with others for Various FPGA Scaling Devices

From the Simulation results, the proposed method is well suited for Large scaled FPGA devices which are widely used in industries now-a-days

IX CONCLUSION

Advanced features of Reconfigurable FPGA devices allow us to design and implement the real time applications in an fast and efficient manner. In this paper, new method of fast and easiest first fit mode of FPGA Placement method using ‘Windowing OR Logic’ method is proposed and its performance is discussed with help of Simulation results. By maintaining less than or equal values of task rejection ratio, the proposed method outperforms in terms of Speed of Execution of an

application. To improve the placement quality, an algorithm with less task rejection ratio has to be developed in future.

REFERENCES

- [1]. Kirash Bazargan, Ryan Kastner and Majid Sarrafzadeh, "Fast Template Placement For Reconfigurable Computing System", IEEE Design and Test of computers, Vol, 17, No. 1, Jan-Mar 2000, pp. 68-83.
- [2] Herbert Walder and Marco Platzner, " Non-Pre Emptive Multitasking On FPGA's: Task Placement And Foot Print Transform, proceedings of the 2nd International Conference On Engineering Of Reconfigurable Systems And Architectures (ERSA), CSREA press, June 2002, pp. 24-30
- [3] Christophe Steger, Herbert Walder, Marco Platzner, Lothar Thielie, "Online Scheduling and Placement of Real Time Tasks to Partially Reconfigurable Devices, Real-Time Systems Symposium (TRSS), Dec 2003, pp. 224-225.
- [4] Herbert Walder and Marco Platzner, "Online Scheduling For Block Partitioned Reconfigurable Devices", IEEE 2003. (DATE 2003, 290-295), pp. 1530-1591.
- [5] A. Ahmadinia & J. Teich, "Speeding Up On-Line Placement for XILINX FPGAs by Reducing Configuration Overhead", Proceeding of IFIP International Conference On VLSI-Soc, Pp. 118-122, December 2003.
- [6] Christoph Steiger, Herbert Walder and Marco Platzner, "Heuristic For Online Scheduling Real Time Tasks To Partially Reconfigurable Devices", FPL 2003 (Proceedings) LNCS, Vol. 2778, pp. 575-584 ©Springer, Heidelberg (2003).
- [7] M. ToMono, M. Nakanishi, S. Yamashita, K. Nakajima and K. Watanabe, "A New Approach To Online FPGA Placement", Proceedings of Conference Of Information Science & System (CISS), April 2004, pp. 145- 150.
- [8] A. Ahmadinia, M. Bednara, C. Bobda & J. Teich, "A New Approach For Online Placement on Reconfigurable Devices", Proceedings Of International Parallel Distributed Processing Symposium (IPDPS), April 2004, pp. 134-140.
- [9] Ali Ahmadinia, Christophe Bobda and Jurgen Teich, "A Dynamic Scheduling And Placement Algorithm For Reconfigurable Hardware", Architecture Of Computing System (ARCS), 2004 ©Springer-Verlag, pp. 125-139.
- [10] Handa and R. Vemuri, " An Efficient Algorithm For Finding Empty Space For Online FPGA Placement", Proceedings DAC, June 2004, pp. 960- 965
- [11] Jin Cui, Zonghua Gu, Weichen Liu and Qing Xu Deng, "An Efficient Algorithm For Online Soft Real time Task Placement On Reconfigurable Hardware Devices", IEEE ISORC, May 2007, pp. 321-328.
- [12] Jin Cui, Qing Xu Deng, Xiuqiang He, Zonghua Gu, "An Efficient Algorithm For Online Management Of 2D Area Of Partially Reconfigurable FPGAs, Design Automation & Test in Europe (DATE), April 2007, pp. 129-134.
- [13] A. Elfrang, H. M. El-Baghdadi & S. I. Shaheen, 'Miss Ratio Improvement For Real Time Application Using Fragmentation Aware Placement', Proceedings, RAW'07, Reconfigurable Architecture Workshop) 2007. [14] X. Zhou, Y.

- wang, X. Huang & C. Peng, "Fast On-Line Task Placement & Scheduling On Reconfigurable Devices", Proceedings, (FPL), 2007, pp132-138
- [15] Thomas Marconi, YiLu, Koen Bertels and Georgi Gaydadjiev, "Online Hardware Task Scheduling and Placement Algorithm on Partially Reconfigurable Devices", ARC 2008, LCIIVS 4943, © Springer-Verlag Berlin Heidelberg 2008, pp. 306-311.
- [16] Thomas Marconi, YiLu, Koen Bertels, GeorgiGaydadjiev, "Intelligent Merging Online Task Placement Algorithm For Partial Reconfigurable System", 2008 EDAA, DATE '08.
- [17] Jesus Tabero, Julio Septin, Hortnsia Mecha & Daniel Mazos, "Allocation Heuristics and Defragmentation Measures for Reconfigurable Systems Management, Integration, The VLSI Journal 2008, pp. 281-296.
- [18] J. septien, D. mozos, H. Mecha, J. Tabero &M. A. Garcia de Dios, "Perimeter Quadrature- Bases Metric For Estimating FPGA Fragmentation In 2D HW Multicasting", Proceedings 2008
- [19] Mostafa Elbidweihy, "Task scheduling and placement algorithm for reconfigurable devices", dissertation, May 2009, Department of Computer electrical.
- [20] Y. Xiao, Z. Duan and P. Nie, "An Efficient Algorithm For Finding Empty Space For Reconfigurable Systems, Proceedings 2009, Third IEEE International Symposium On Theoretical Aspects Of Software Engineering, pp. 36-43, 2009.
- [21] Trong-Yen Lee, Che-Cheng Hu and Chia-ChunTsai, "Adaptive Free Space Management Of Online Placement For Reconfigurable Systems, Proceedings of International Multi Conference of Engineers & Computers Scientists, IMECS 2010, Hong Kong. Vol. 1, March 2010, pp. 17-19.
- [22] Chuan Hong, Khaled Benkrid, Xabier Turbe, Ahmet T. Erolgen, Tughrul Arslan, "An FPGA Task Alloator With Preliminary First- Fit 2D Packing Algorithms", 2011 NASA/ESA Conference On Adaptive Hardware & Systems(AHS-2011) Y
- [23] Yi Lu, Thomas Marconi, Georgi Gaydadjiev, Koen Bertels and Roel Meeuws, " A Self Adaptive On-Line Task Placement Algorithm For Partially Reconfigurable Systems.
- [24] Manish Handa and Ranga Vemuri, " Area Fragmentation in Reconfigurable Operating Systems", Engineering of Reconfigurable Systems and Algorithms Las Vegas, NV, June 2004. Department of ECECS, University of Cincinnati, pp 77-83.
- [25] Maisam Mansub Bassiri and Hadi Shahriar Shahhoseini, "Configuration Reusing in Scheduling for reconfigurable Computing System", Journal of Computer Science and Technology, 26(3):463-473 May 2011. DOI: 10.1007/s11390-011-11472

AUTHORS

B. Premalatha received B. E Degree in ECE From Kumaraguru College of Technology, Coimbatore, Tamilnadu in 2000 and received her ME degree in VLSI Degree from Anna university, Chennai Guindy campus, Tamilnadu, India in 2007. , She is awarded Gold Medal for academic Performance from Anna University in 2007. She is Pursuing PhD in Faculty of Information and communication engineering Anna University, Chennai, Tamilnadu, India. Her current research interest includes CAD Algorithms in VLSI, Low power VLSI and Reconfigurable Computing. She is an Life Member of ISTE(India).



Dr. S. UmaMaheswari received her B. E Degree in Electronics and Communication Engineering from Government College of Technology, Coimbatore in the year 1985 and M. E (Applied Electronics) from Bharathiar University in 1991. She received her Ph. D degree in the area of Biometrics from Bharathiar University, Coimbatore in the year 2009. She is Associate Professor of Electronics and Communication Engineering department in Coimbatore Institute of Technology. She is having more than 26 years of teaching experience. She has published technical papers in national /international conferences/ journals. Her special fields of interest are Digital Image Processing and Digital Signal Processing. She is a Member of IE (India), Life Member in Indian Society for Technical Education (India), Life Member in Systems Society of India, and Life Member in Council of Engineers (India).