Performance Driven Floorplanning for FPGA Based Designs*

Jianzhong Shi and Dinesh Bhatia
Design Automation Laboratory
P.O. Box 210030, ECECS Department
University of Cincinnati
Cincinnati, OH 45221-0030

ABSTRACT

Increasing design densities on large FPGAs and greater demand for performance, has called for special purpose tools like floorplanner, performance driven router, and more. In this paper we present a floorplanning based design mapping solution that is capable of mapping macro cell based designs as well as hierarchical designs on FPGAs. The mapping solution has been tested extensively on a large collection of designs. We not only outperform state of the art CAE tools from industry in terms of execution time but also achieve much better performance in terms of timing. These methods are especially suitable for mapping designs on very large FPGAs.

I INTRODUCTION

Field programmable gate arrays are becoming extremely popular for implementing small and medium scale designs. CAD tools for automatic placement and routing for physical design have been developed recently. However, increasing design densities on large FPGAs and greater demand for performance, has called for special purpose tools like floorplanner, performance driven router, and more. Recently, Xilinx has started to supply a floorplanner with the latest release of XACT tools, which is particularly useful for structured designs and data path logic [12]. While it is a good design aid, the floorplanner is completely manual in its execution. Motivated with the task of automating the floorplanning for FPGAs, we have targeted this research for floorplanning.

The macro blocks are often called RPMs (Relationally Placed Macros). Every macro contains several circuit elements connected together to realize a particular function. For example different types of gates can be connected together to realize functions like multiplication, addition, and more. Each macro block contains several CLBs that realize its logic functionality. Figure 1 illustrates the FPGA physical design flow under this new methodology. The input of macro block based FPGA floorplanning is a set of macro blocks and a netlist defines their interconnection. In the floorplanning, the macros are assigned specific shapes and locations on the FPGA die. After floorplanning, the CLBs within the reshaped macros are placed for complete assignment. A constraint file defining the complete placement is then generated for carrying out the next task, i.e., routing. In this paper we present a floorplanning based design mapping solution that is capable of mapping macro cell based designs as well as hierarchical designs on FPGAs.

The rest of the paper is organized as follows. Section II gives a brief literature review of related research on physical VLSI floorplanning. Section III describes the preliminary notations, definitions, and floorplanning. Section IV gives an overview of experimental setup followed by section V that gives results and analysis.

II RELATED RESEARCH

Floorplanning is an important stage in the physical design of custom VLSI circuits. Sastry and Parker showed that the two-dimensional floorplan compaction problem is \textit{NP-complete} [5]. Stockmeyer showed that area minimization for general floorplans is strongly \textit{NP-complete} [7]. Various strategies have been reported in the literature [3] [8] [9] [11]. Physical VLSI floorplans can be broadly classified as constructive and iterative [6]. Some of the constructive techniques use duster growth, partitioning and slicing, connectivity clustering, and rectangular dualization [6] [11]. Other techniques include constraint graph based, mathematical programming, or knowledge-based approaches.

Iterative and non-deterministic techniques such as simulated annealing have also been reported to solve the

*supported in part by contract number F33615-95-C-1715 from US Air Force.
physical VLSI floorplanning problem[6]. There are no floorplanning algorithms that are targeted towards FPGAs.

In [4], a procedure for path-delay constrained initial placement is presented which directly incorporates timing and geometrical constraints. The problem is modeled and mathematically formulated as a constrained nonlinear programming problem. In [13], a floorplanning algorithm that uses a force-directed approach is presented. The best floorplan with timing optimization is selected, and overlaps are removed by applying constraint reduction and general cell reshaping [13]. Due to the shape restriction of FPGA macro blocks and CLB resources, special treatment should be considered during each FPGA floorplanning step.

III Floorplanning Based Physical Design

In our approach to hierarchical design, two major steps are involved. The first step, namely floorplanning, produces a macro level layout design on large FPGAs. Floorplanning is followed by detailed placement within the macro blocks. In our current implementation, we make use of simulated annealing based detailed placement. Although we have mentioned macro block level floorplanning, we have been able to demonstrate that floorplanning using hierarchical blocks is feasible at any level of design hierarchy.

A FPGA Floorplanning

We first describe some terms and floorplanning problem.

- **Macro Blocks:**
  Macro block based FPGA design implements functional block units on the FPGA chip. Each functional block contains one or more CLBs and realizes a small logic function. The functional blocks can also be pre-defined macro blocks maintained in a macro library. The macro block development process accepts the macro requirements from the high level synthesis tool. Once the macro block I/O and function are determined, the macro size is estimated and locations are assigned to the individual CLBs inside the macro. The Macro blocks may have fixed or flexible rectangular shapes. The predefined macros are sorted by their functions in the macro library. These macros are then used to make a design.

- **Input of FPGA Floorplanning:**
  The input for FPGA floorplanning problem can be formulated as follows:

1. A set of \( m \) rectangular blocks \( B = \{b_1, b_2, \ldots, b_m\} \). For each \( b_i \in B, 1 \leq i \leq m, w_i, h_i \) are the width and the height of \( b_i \). \( w_i \) and \( h_i \) are the number of CLBs in the width and the height of a macro block. For macro blocks where (due to performance reasons) a pre-defined shape (geometry) is essential, the \( w_i \) and \( h_i, j \in \{1, 2, \ldots, m\}, \) are fixed constants. We also associate \( a_i \) with each \( b_i, 1 \leq i \leq m, \) as an area of \( b_i \) (i.e., \( a_i = w_i \times h_i \)).
2. A set of nets \( N = \{n_1, n_2, \ldots, n_K\} \) describing the connectivity information among blocks.

3. Timing constraints or architectural design specification for ASICs, such as the logic data path or the pipeline design.
4. FPGA chip dimensions, usually a square array.

- **Output of FPGA Floorplanning:**
  A legal floorplan, i.e., a floorplan satisfying the following constraints and objectives:

1. Each macro/functional block \( b_i \) is assigned to a location \((x_i, y_i)\) on the FPGA chip;
2. No overlapping between two blocks;
3. Fit all blocks on the fixed (area and dimension) FPGA chip;
4. Each CLB inside the macro blocks is assigned a physical location on the FPGA chip;
5. Satisfy the pre-located I/O pads constraints.

A-1 Topological Placement

In this step, we treat all macro blocks as topological points without considering their real physical shape and size, and construct relative topological relations among the macro blocks which is optimal for wiring, timing, and other design constraints. This relative placement step can result in an overlapping floorplan, i.e., a floorplan with block overlaps.

We obtain topological plan using a force directed placement or by considering the architectural features of the design. Our approach for the former is similar to the one in [13]; however, for latter, we consider the data flow in the datapath or the pipeline. Using this information of data flow, a topological ordering is built which is then inserted into the FPGA architecture. The topological floorplan is then modified such that the design constraints for wirelength are minimized while topological data flow is honored.

A-2 Force Directed Topological Arrangement Approach

This approach explores the similarity between the topological problem and classical mechanics problem of a system of bodies attached to springs. In this approach, the blocks connected to each other by nets are supposed to exert attractive forces on each other. The connectivity-based force is proportional to the number of connections between blocks. Thus, highly connected blocks will be placed close to each other. The exerted connectivity-based force on block \( b_i \) due to block \( b_j \) is given by \( F_{ij} = c_{ij} \), where \( c_{ij} \) represent the number of connections between blocks \( b_i \) and \( b_j \). The zero-force connectivity sensitive location is given by

\[
\begin{align*}
x_i^* &= \frac{\sum_{j=1}^{n} p_{ij} F_{ij} x_j}{\sum_{j=1}^{n} p_{ij} F_{ij}}, \\
y_i^* &= \frac{\sum_{j=1}^{n} p_{ij} F_{ij} y_j}{\sum_{j=1}^{n} p_{ij} F_{ij}}
\end{align*}
\]

In addition, the nets with critical timing should be able to define the cost (force) due to timing requirements. In practice, the net weighting is done manually by the designer. However, there are few known quantitative methods that will compute the delay bounds for weighting each net. One such method has been proposed by Nair[2]. This method computes slack on the delay paths and distributes them uniformly over the nets belonging to the path. The
slack is defined as the difference between the pre-defined worst case delay and the actual delay (due to modules only) in a path. Clearly, higher is the positive slack, less critical are the nets belonging to the path. In other words, the net weight computed by Nair’s method[2] is inversely proportional to the criticality of the net. Thus, timing force $F_i^t$ exerted by net $i$ is given by,

$$F_i^t = \frac{t_{CLK} - d_i}{t_{CLK}}$$  \hspace{1cm} (2)$$

where, $t_{CLK} = 1/f$ and $f$ is the desired frequency of operation, $d_i$ is the criticality associated with net $i$.

The zero-force timing sensitive location is given by

$$x_i^s = \frac{\sum_{j=1}^n p_{si} F_j^t x_{si}}{\sum_{j=1}^n p_{si} F_j^t}, \hspace{1cm} y_i^s = \frac{\sum_{j=1}^n p_{si} F_j^t y_{si}}{\sum_{j=1}^n p_{si} F_j^t}$$  \hspace{1cm} (3)$$

In equations 1 and 3, $p_{si}$ is the percentage of placed blocks belonging to the net connecting $b_i$ and $p_{si}$.

With zero-force sensitive locations of connectivity and timing, the final topological point target location is defined as

$$x_s = a_1 x_i^s + a_2 x_i^s, \hspace{1cm} y_s = a_1 y_i^s + a_2 y_i^s$$  \hspace{1cm} (4)$$

where, $0 \leq a_1, a_2 \leq 1$, and $a_1 + a_2 = 1$.

The force directed topological arrangement is obtained in a constructive fashion. A block with maximum gain is greedily moved each time to a suitable location. The gain calculation for block $b_i$ is defined as,

$$G_i = \sum_{j \in PF_b} c_{ij} + \beta \sum_{j \in N_b} (1 - p_j) cost_j$$

where, $PF_b$, $1 \leq k \leq m$, is set of blocks in the partial floorplan at step $k$ of floorplanning; $B_j$ is set of blocks interconnected by net $n_j$; $N_b$ is set of partial nets at step $k$ of the floorplanning process; $p_j$ is percentage of placed blocks of $B_j$, where $0 \leq p_j \leq 1$, $c_{ij}$ is connectivity of block $b_i$ to block $b_j \in PF_b$; $\beta$ is real positive weight coefficient.

Three sets will be employed and updated during this step: the placed set which contains the already positioned blocks, the adjacent set which contains the blocks having common connections with the element in the placed set, and the unplaced set containing the remaining blocks which do not belong to either the placed set or the adjacent set. Initially, all pre-located I/O pads are put in the placed set. All blocks having connections with these I/O pads are in the adjacent set, and the rest of the blocks are in the unplaced set.

The topological arrangement algorithm starts by selecting the first seed block. For the pre-located I/O pads problem, we put these pads on the center of the FPGA chip boundaries and assume the chip dimensions to be sufficiently large. The block with the maximum gain is chosen.

**Topological Placement Approach:** Due to the special characteristics of the FPGA floorplanning, we need to do placement for the topological arrangement results according to FPGA chip dimension. This step is necessary in order to increase the final CLB utilization. The placement approach minimizes the layout area while keeping the topological relations among the modules, i.e., two modules with close relation in the topological arrangement are kept close together in the placement. The topological placement algorithm performs in two phases: phase1 places all macro blocks in one row and phase2 folds one row to obtain a multi-row design to minimize the area or satisfy the layout aspect ratio. These two phases can be seen in most of the classical placement or estimation methods [1]. The detailed steps in the two phases are given below:

**A. Phase 1:**

In this phase a one-row design is constructed in the following steps.

- step1: choose the rightmost-bottom topological point as the starting point in a sequential list.
- step2: find the next topological point not belonging to the sequential list and having the shortest distance from the starting point. This point (module) is defined as the first heavily connected point. Intuitively speaking, the first heavily connected point has the largest number of connections with the starting point or the signals connecting these two points belong to the critical path.
- step3: put the first heavily connected point into the sequential list and assign this point as a new starting point.
- step4: repeat step 2 - step 3 until all modules are in the sequential list.

For logic datapath design, we can create this sequential list directly by reading the input design datapath file.

**B. Phase 2:**

In this phase the one row design from the sequential list is folded into a two dimensional practical layout. Phase 2 proceeds in the following steps:

- step1: assign the initial ideal left space parameter(ils). This parameter shows the difference between the folded incremental length of one row design and the real FPGA chip dimension. When the incremental length exceeds ILS, we stop the one row folding.
- step2: construct one row placement after folding. The location of each module is represented by $M[row][column]$. We start row folding at $row = 1$, and for the next row iteration $row = row + 1$. For each folding row, column starts at $column = 1$, and for the next column iteration $column = column + 1$.
- step3: sort and swap orientation of column values for each module with the same even row number.
- step4: compute the critical(longest) path of the current placement result.
- step5: increase the ILS by an integer rate, typically by 1.
- step6: repeat step 1 - step 5 by given iterations and find the minimized area topological placement result.

The output from the topological placement is a list of the FPGA macro blocks with the topological point xy-coordinates and a list of the I/O pads with their assigned boundary locations on the FPGA chip.
A-3  The Legal Floorplan

In this phase we need to construct a legal floorplan which respects the topological constraints between the macro blocks which are obtained during topological arrangement step. Two directed acyclic graphs (DAG) are employed to model the topological constraints. The topological relations of the macro blocks are translated into a horizontal constraint graph $G_H$ and a vertical constraint graph $G_V$ according to their $xy-$coordinates on the FPGA die. The vertex set of $G_H$ is a set of macro blocks plus two dummy vertices, L and R, which correspond to the left and right boundaries of the FPGA chip. Similarly, the vertex set of $G_V$ is the set of macro blocks plus two dummy vertices, T and B, which correspond to the top and bottom boundaries of the FPGA chip. The edge set of $G_H$ models the to-the-left/to-the-right topological relationships of the macro blocks on the FPGA die, while that of $G_V$ models the on-the-top/on-the-bottom topological relationships of the macro blocks on the FPGA die. In other words, let $G_H = (V, E), V = \{ b_1, b_2, \ldots, b_n, L, R \}$. Vertex $L$ has in-degree equal to zero and vertex $R$ has out-degree equal to zero. For any two vertices $v_i$ and $v_j$ such that $v_i$ is to the left of $v_j$ in the topological floorplan, then there exists a directed edge $(v_i \rightarrow v_j) \in E$. Similarly, $G_V$ is built with top to bottom relationships.

We translate the topological relationship of the macro blocks to a complete constraint set in the form of $G_H$ and $G_V$. A floorplan that satisfies a complete constraint set is legal, i.e., no overlaps. In order to get high CLB utilization on FPGA floorplan, the constraint set of two graphs $G_H$ and $G_V$ should be constructed by removing all redundant constraints. This method is similar to the algorithm used in [15], and in contrast to the algorithm in [10].

The legal floorplan algorithm is as follows:

- First, the algorithm examines each pair of macro blocks in $B$ set (set of macro blocks) and inserts topological constraints in $G_H$ or $G_V$ according to their $xy-$coordinates.
- If the $xy-$coordinates of the two blocks $i$ and $j$ are different, these two blocks are said to be constrained in both horizontal and vertical directions. In this case an edge $(i, j)$ is inserted in each of $G_H$ and $G_V$.
- The algorithm calculates the longest path that goes through the inserted edge $(i, j)$ in $G_H$ and $G_V$.
- The edge that yields the shorter path in the current graph is retained and the other is removed.
- If either $x-$coordinates or $y-$coordinates of the two blocks $i$ and $j$ are equal, these two blocks are said to be constrained in only one direction. There is only one choice in this case and therefore the algorithm inserts the edge in the corresponding graph.

This approach removes all redundant constraints between any pairs of the macro blocks from the constraint set. This produces a more compact legal FPGA floorplan result. The next reshaping step reshapes the soft macro blocks in order to optimize the legal floorplan area. Soft macros are ones for which overall shape is not important for performance evaluation and characterization.

A-4 Reshaping of Soft Macro

The reshaping algorithm uses the constraint graphs $G_H$ and $G_V$ to compute the dimensions of the soft macro blocks and final floorplan area. The algorithm iteratively reshapes only one soft macro block each time on the longest path. The dimension of each soft block on the selected critical path is reduced by a user-specified factor (see Figure 2). Our algorithm only reshapes one selected block on the critical path. This is consistent with specific characteristics of macro block reshaping as shown in Figure 2. During the reshaping iterations the set of shapes, corresponding to the least area floorplan, is stored and updated. These shapes are then used as input for the next path of the algorithm. Note that the constraint set remains unchanged during the reshaping step.

The overview of the reshaping algorithm is given below:

```
Algorithm BlkReshp()
{
    For j := 1 to number of reshape iterations do
    {
        Store the current shapes of blocks
        Let $P$ be most critical path of $V$ graph
        or $H$ graph
        if $P < DieDimension$, EXIT
        else
        {
            Reshape one soft macro by
            one column or row on $P$
            if $P$ is shorter than before
            {
                store the current shape of
                the block
            }
            do reshaping for next soft macro
        }
        else
        {
            Do next soft macro until its shape can be
            changed
        }
    }
    j++
}
```

Fig. 2. Reshape soft macro block by factor of 1. Suppose (a) represent a 8-CLB macro block. A compression by a factor of 1 in length results in (b). A compress by a factor of 1 in height on macro in (a) results in (c).
IV OVERVIEW OF CAD ENVIRONMENT

At the beginning, the FPGA floorplanning step takes an input in the form of a netlist which describes the connectivity between various macros. For hierarchical designs, this will be a netlist representing higher level interconnection among the design modules. Upon completion of floorplanning, a constraint file is generated. The constraint file contains the physical mapping information for each macro. For macro based designs, we maintain one hand crafted physical implementation of each macro in a library. This physical implementation is rigid in shape and aspect ratio. Thus, after floorplanning, the macros for which the shape does not differ (remains same) from library implementation, a CLB based detailed placement is inserted from the library. For the macros which are reshaped, a local placement of CLBs within the macro with the new shape requirements is performed.

To effectively illustrate the advantage of floorplanning and hierarchical designs, we hand crafted six benchmark examples using ViewLogic ViewDraw tool. These examples were mapped on Xilinx 4000 and 4000E family of devices.

A Benchmark Description

Six macro block based benchmark examples were designed to test our hierarchical approach. MFilter is a morphological filter example which is used in digital signal processing. DiErsn is an extension of MFilter example. It contains dilation and erosion part of the morphological filter. Mult-10 and Mult-16 are 10 and 16 bit multipliers respectively. They are purely combinational successive addition type multipliers. CLA is an 128-bit complete carry-lookahead tree adder which is used in computer arithmetic area. CLA-D is functionally the same as CLA but with different physical structure in terms of macro block construction.

Table I gives some of physical characteristics for the designs in terms of the number of macros, number of CLBs, number of Input/Output blocks and number of nets. It should be noted that the macro blocks illustrated in the high level diagram also form the macros that are floorplanned in the design. The MFilter design utilizes following macro types:

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Macros</th>
<th>Macro Types</th>
<th>CLBs</th>
<th>Nets</th>
<th>I/OBs</th>
</tr>
</thead>
<tbody>
<tr>
<td>MFilter</td>
<td>17</td>
<td>3</td>
<td>157</td>
<td>320</td>
<td>32</td>
</tr>
<tr>
<td>DiErsn</td>
<td>34</td>
<td>3</td>
<td>314</td>
<td>821</td>
<td>32</td>
</tr>
<tr>
<td>Mult-10</td>
<td>10</td>
<td>1</td>
<td>200</td>
<td>171</td>
<td>41</td>
</tr>
<tr>
<td>Mult-16</td>
<td>16</td>
<td>1</td>
<td>312</td>
<td>400</td>
<td>65</td>
</tr>
<tr>
<td>CLA</td>
<td>128</td>
<td>3</td>
<td>448</td>
<td>1157</td>
<td>135</td>
</tr>
<tr>
<td>CLA-D</td>
<td>32</td>
<td>3</td>
<td>576</td>
<td>397</td>
<td>133</td>
</tr>
</tbody>
</table>

TABLE I
Characteristics of preliminary benchmarks

V RESULTS AND ANALYSIS

In order to compare our hierarchical approach against the traditional FPGA design approach of Xilinx PPR (XACT5.2) tool, we also implemented all designs in two different ways using Xilinx PPR tools. One way is to map the design such that it has the same physical structure as our hierarchical floorplanner. In this representation, the mapping of design on function generators and flip flops is identical to the one we used for floorplanning. The other way is to allow the Xilinx PPR tool to do a totally flattened design mapping which results in a different physical structure of the design.

The performance of our approach is evaluated in terms of performance parameters like the CPU execution time, device utilization, routability and final path delay of the routed design. The comparison of the corresponding results from Xilinx PPR tool has also been made.

A CPU Execution time

We measured CPU execution time for mapping all benchmark circuits on different FPGA devices using the floorplanner against Xilinx XACT tool. The experiments have been done on a Sun Sparc5. Table II summarizes the results of relevant execution time for the six benchmark circuits. For our hierarchical floorplan and placement approach, the CPU time for macro level floorplanning of each design is listed in the second column. The third column gives the CPU time for local placement if the shape of the macro block has been changed during floorplanning step or, if we have to use Xilinx PPR tool to create the placed design of the user defined macros. The total execution time for physical mapping of a design is the sum total of the floorplanning time, and local placement time. This data is listed in fourth column. The CPU time for physical mapping has also been measured using Xilinx XACT tool for each design. In order to make a meaningful comparison with our floorplanning results, we measured CPU execution time under two different mapping conditions, i.e., the same physical structure mapping, and flattened design mapping for each of the benchmark circuit. For the first one, we force XACT tool to do physical mapping under the same number of function generators and D flip-flops as used in our floorplanning tool. This will make the floorplanning tool and XACT placement tool do the physical mapping under the same physical structure for a given FPGA design input. For the flattened design mapping, we let the XACT tool do a totally flattened design for each circuit. This leads to a different number of function generators and D flip-flops for the logic design input. For Xilinx XACT placement tool, we also measured CPU execution time using two different placement
efforts. One is the default value and one is maximum value. The placement effort represents the hardness in the timing driven mode for doing FPGA placement. The corresponding CPU execution time for the same physical structure of the design using default placement effort and maximum effort is given in the fifth and sixth column of table II. The seventh and eighth column in table II list the CPU time for the flattened design mapping using default placement effort and maximum effort respectively.

The CPU execution results show that our hierarchical floorplan and placement approach reduces the physical mapping time significantly in most cases. As discussed previously, the total number of macro blocks to be floorplanned is always much less than the total number of CLBs to be placed for the same functional design. This is one of the advantages for doing FPGA floorplanning. Furthermore, in most macro block based designs, the total number of macro blocks is between 20 and 40.

B Routability

Routability of a circuit is an important criterion to assess the quality of a given FPGA floorplan and placement. The routability metric is evaluated in terms of the percentage of the routed nets obtained after routing the circuit. The CPU time for routing and final routability of each mapped design have been measured using the Xilinx XACT router. The results are given in Table IV and Table V. The mapped designs for each circuit are obtained using the floorplan tool and the XACT tool separately. We also obtained two kinds of mapping results using the XACT tool. One is for the same physical structure mapping, and the other one is for flattened design mapping. The CPU time for routing the placement results for the same physical structure of the design using the default placement effort and the maximum placement effort is listed in the column 5 and 6 of table IV respectively. Column 7 and 8 of table IV give the CPU time for routing the placement results for the flattened design using the default placement effort and the maximum placement effort respectively.

In terms of routability, our floorplanned design results in none unrouteable nets for all the circuits. However, the XACT flattened design results in two unrouteable designs for Mult16 and CLA which are placed using the default placement effort. Furthermore, in most cases the the time to perform complete routing is much less for designs which were floorplanned when compared to the designs that were placed using Xilinx XACT tools.

C Performance Measurement

The final performance of the six benchmark circuits were measured using \texttt{xdelay} static timing analyzer which is part of the Xilinx XACT tool. The tool gives the average delay obtained after routing would be a near true estimate since it would incorporate the delay values due to the presence of switch boxes. The results shown in Table VI reflect the maximum clock frequency for floorplan and XACT placement of the design circuits respectively. On the average, we have observed about 10% improvement over the results obtained using the Xilinx XACT placement tool to do the same physical structure design, and 30% improvement over the flattened design.

### Table III

<table>
<thead>
<tr>
<th>Circuit</th>
<th>A</th>
<th>B</th>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mult16</td>
<td>20.4</td>
<td>24.5</td>
<td>XC4013</td>
</tr>
<tr>
<td>Mult16</td>
<td>31.1</td>
<td>34.2</td>
<td>XC402SE</td>
</tr>
<tr>
<td>CLA</td>
<td>6.7</td>
<td>8.6</td>
<td>XC402SE</td>
</tr>
<tr>
<td>CLA-D</td>
<td>3.5</td>
<td>5.3</td>
<td>XC402SE</td>
</tr>
</tbody>
</table>

### D Timing Constraint Testing

The FPGA floorplanner has been tested with timing constraints. Due to the lack of a macro block based FPGA timing analyzer, we manually assign the upper bound delay of the nets. The most critical path can be specified by pre-running Xilinx \texttt{xdelay} timing analyzer which can sort the paths according to their criticality in the design. The nets of the most critical path are assigned small delay values (expressed in nano-seconds). The net costs are then used to guide the floorplanner to do topological arrangement. Three benchmark circuits have been tested using this method and the final performance results obtained from \texttt{xdelay} are shown in Table III.

### VI Conclusions

In this work we have presented a methodology for hierarchical floorplanning and placement for large FPGA based designs. As integration densities increase, the need for CAD tools for quick and efficient mapping of designs on FPGAs is increasing. Our approach provides one such solution. For performance oriented macro based designs, FPGA floorplanning is very helpful as custom hand crafted small macros are easy to design. We have made the best possible effort to generate large benchmark examples which are available for community use. Our approach is successful in its implementation and can greatly cut down the design time. Future work in this area would include floorplanning with emphasis on FPGA routability and hierarchical routing.

### References


TABLE II
Comparison of CPU time for mapping a design using floorplanner against Xilinx XACT tool

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Floorplan</th>
<th>PPR Placement</th>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>MFilter</td>
<td>313</td>
<td>479</td>
<td>N/A</td>
</tr>
<tr>
<td>MFilter</td>
<td>463</td>
<td>498</td>
<td>N/A</td>
</tr>
<tr>
<td>MFilter</td>
<td>543</td>
<td>414</td>
<td>N/A</td>
</tr>
<tr>
<td>DiEsF</td>
<td>3186</td>
<td>2916</td>
<td>N/A</td>
</tr>
<tr>
<td>Multi10</td>
<td>166</td>
<td>281</td>
<td>N/A</td>
</tr>
<tr>
<td>Multi8</td>
<td>1868</td>
<td>2635</td>
<td>5hrs</td>
</tr>
<tr>
<td>Multi8</td>
<td>3256</td>
<td>4537</td>
<td>4hrs</td>
</tr>
<tr>
<td>CLA</td>
<td>872</td>
<td>291</td>
<td>12hrs</td>
</tr>
<tr>
<td>CLA-D</td>
<td>245</td>
<td>318</td>
<td>382</td>
</tr>
</tbody>
</table>

TABLE III
CPU time for routing by using Xilinx XACT router

<table>
<thead>
<tr>
<th>Circuit</th>
<th>CPU time for routing (sec)</th>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>MFilter</td>
<td>5</td>
<td>N/A</td>
</tr>
<tr>
<td>MFilter</td>
<td>4</td>
<td>N/A</td>
</tr>
<tr>
<td>MFilter</td>
<td>4</td>
<td>N/A</td>
</tr>
<tr>
<td>DiEsF</td>
<td>96</td>
<td>N/A</td>
</tr>
<tr>
<td>Multi10</td>
<td>36</td>
<td>N/A</td>
</tr>
<tr>
<td>Multi8</td>
<td>40</td>
<td>4hrs</td>
</tr>
<tr>
<td>CLA</td>
<td>3hrs</td>
<td>12</td>
</tr>
<tr>
<td>CLA-D</td>
<td>120</td>
<td>141</td>
</tr>
</tbody>
</table>

TABLE IV
CPU time for floorplanning and FPGA device

<table>
<thead>
<tr>
<th>Circuit</th>
<th>CPU time of floorplan</th>
<th>CPU time of PPR</th>
<th>FPGA device</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Floorplan</td>
<td>MacroPlace</td>
<td>Total</td>
</tr>
<tr>
<td>MFilter</td>
<td>313</td>
<td>479</td>
<td>N/A</td>
</tr>
<tr>
<td>MFilter</td>
<td>463</td>
<td>498</td>
<td>N/A</td>
</tr>
<tr>
<td>MFilter</td>
<td>543</td>
<td>414</td>
<td>N/A</td>
</tr>
<tr>
<td>DiEsF</td>
<td>3186</td>
<td>2916</td>
<td>N/A</td>
</tr>
<tr>
<td>Multi10</td>
<td>166</td>
<td>281</td>
<td>162</td>
</tr>
<tr>
<td>Multi8</td>
<td>1868</td>
<td>2635</td>
<td>5hrs</td>
</tr>
<tr>
<td>Multi8</td>
<td>3256</td>
<td>4537</td>
<td>4hrs</td>
</tr>
<tr>
<td>CLA</td>
<td>872</td>
<td>291</td>
<td>12hrs</td>
</tr>
<tr>
<td>CLA-D</td>
<td>245</td>
<td>318</td>
<td>382</td>
</tr>
</tbody>
</table>

TABLE V
Final routability of the mapped designs under Xilinx XACT router

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Final routability (%)</th>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>MFilter</td>
<td>100</td>
<td>N/A</td>
</tr>
<tr>
<td>MFilter</td>
<td>100</td>
<td>N/A</td>
</tr>
<tr>
<td>MFilter</td>
<td>100</td>
<td>N/A</td>
</tr>
<tr>
<td>DiEsF</td>
<td>100</td>
<td>N/A</td>
</tr>
<tr>
<td>Multi10</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Multi8</td>
<td>100</td>
<td>81</td>
</tr>
<tr>
<td>CLA</td>
<td>100</td>
<td>94</td>
</tr>
<tr>
<td>CLA-D</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

TABLE VI
Xilinx XACT static timing analyzer.

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Xilinx XACT</th>
<th>MHz</th>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>MFilter</td>
<td>24.4</td>
<td>13.8</td>
<td>N/A</td>
</tr>
<tr>
<td>MFilter</td>
<td>24.4</td>
<td>13.8</td>
<td>N/A</td>
</tr>
<tr>
<td>DiEsF</td>
<td>28.3</td>
<td>18.3</td>
<td>N/A</td>
</tr>
<tr>
<td>DiEsF</td>
<td>28.3</td>
<td>18.3</td>
<td>N/A</td>
</tr>
<tr>
<td>Multi10</td>
<td>6.7</td>
<td>4.8</td>
<td>3.3</td>
</tr>
<tr>
<td>Multi10</td>
<td>6.7</td>
<td>4.8</td>
<td>3.3</td>
</tr>
<tr>
<td>CLA</td>
<td>6.7</td>
<td>4.8</td>
<td>3.3</td>
</tr>
<tr>
<td>CLA-D</td>
<td>6.7</td>
<td>4.8</td>
<td>3.3</td>
</tr>
</tbody>
</table>


