**IJEMD-CSAI**, 1 (1) (2022), 8-17 https://doi.org/10.54938/ijemdcsai.2022.01.1.80

**International Journal of Emerging Multidiciplinaries:**

**Computer Science & Artificial Intelligence**

***Research Paper***

*Journal Homepage: www.ijemd.com*

*ISSN (print): 2791-0164*

# Evaluating Coarse-Grained FPGAs having Hard-Blocks Placed in Columns

**Husain Parvez**

*LIP6, Université Pierre et Marie Curie 4, Place Jussieu, 75005 Paris, France*

**Abstract**

This work uses a coarse-grained FPGA architecture exploration environment to compare a column-based FPGA architecture with a non-column based architecture. Different groups of netlists are collected and a single floor-planning is optimized for all the netlists in a group. It has been found that the coarse-grained architectures that do not limit their hard-blocks to columns give much better placement costs and routing channel usage than the architectures that limit their hard-blocks to columns. Though a column based architecture can give a more compact layout as compared to a non-column based architecture; but the latter gives higher gains in the channel width requirements and can result in overall area gain.

## 1. Introduction

A number of commercial FPGA architectures [4] [3] have hard-blocks (Multipliers and RAMS) placed in columns. The basic model of such kind of FPGA can be represented by Figure 1.1 where the columns of hard-blocks are placed in between the soft-blocks. Placing the hard-blocks in columns not only facilitates the generation of the final layout but it also permits to change the widths of the complete column according to area needs, eventually generating a compact layout. Whereas a non-column based floor-planning requires both the width and height of the hard-blocks to be multiple of smallest block in the architecture, which causes some area loss. This work shows that the hard-blocks not placed in columns result in better placement cost, and eventually lesser channel width is required to route a netlist, and thus area gain is achieved.
This work uses a new coarse-grained FPGA architecture exploration environment [6]. The major feature of this environment is that it refines the FPGA floor-planning along with the placement of netlist. It can also place multiple netlists and changes their architecture floor-planning to get a trade-off architecture for a given set of netlists. A number of other exploration environments have been proposed and extensively used. Like VPR [8] (Versatile Place and Route) has been extensively used for the exploration of fine grained FPGAs. Inherently it does not support coarse grained blocks. However [7] and [13] have extended VPR to explore specific coarse grained architectures. [10] has developed the virtual embedded block methodology (VEB) to model arbitrary embedded blocks on existing commercial FPGAs. Later [18] has incorporated VEB methodology in VPR, thus enabling the support of architectures other than commercial FPGAs. [16] has also developed a CAD tool for FPGAs with Embedded Hard Cores. In comparison to our environment, all these previous environments propose a pre-determined floor-planning organization and does not consider the problem of finding the block positions in the architecture.
Besides using the new exploration environment [6], this work also upgrades it in many respects. Different parameters and techniques used by VPR [8] are implemented here which has helped improve the results. The parameters of the simulated annealing algorithms are set to be the same as that used by VPR [8]. The Range

***Email addresses*: parvez.husain@lip6.fr (H. Parvez)**

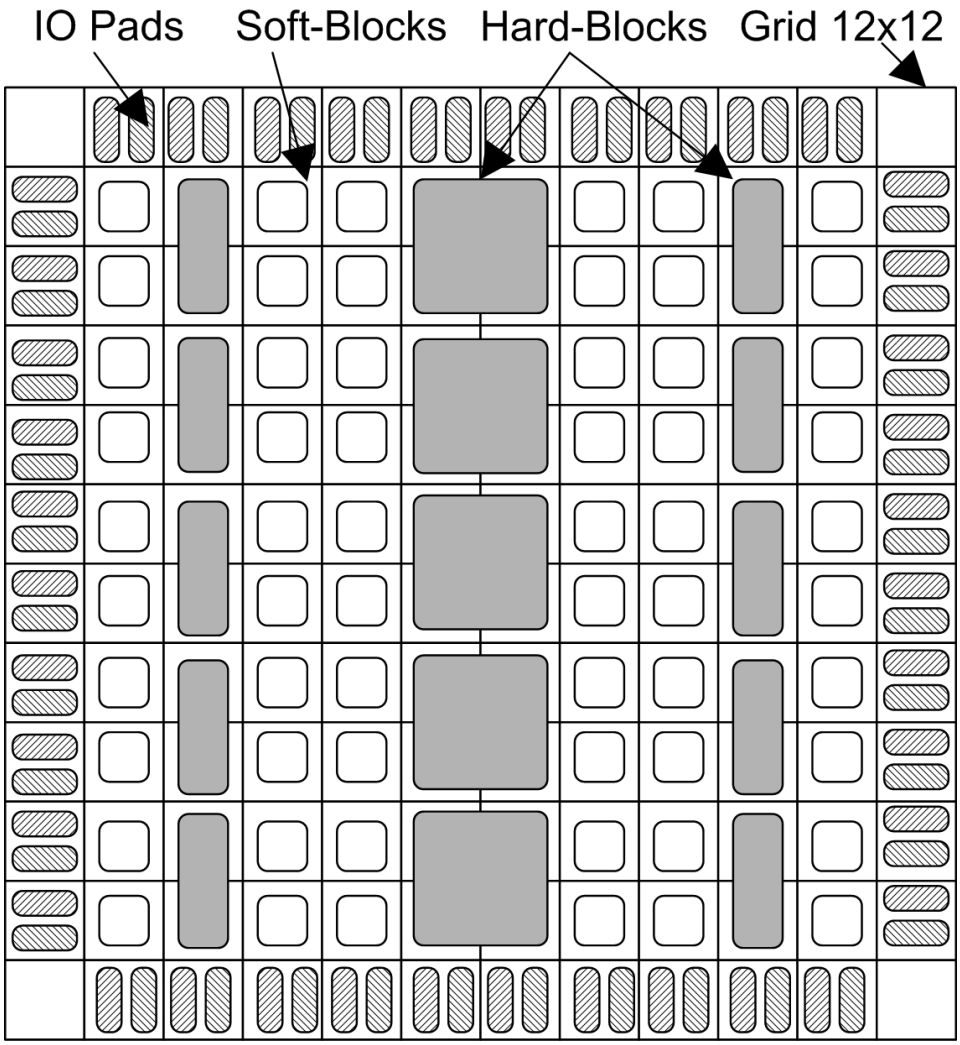



**Figure 1.1: Coarse-grained FPGA**

Limiter operation (discussed later in detail) is also applied. The implementation is further optimized and the execution time information is added in this work. The initial reference architecture used for comparison in the earlier work [6] was a very elementary architecture which does not at all represented any commercial architecture floor-planning. So this work uses the reference architecture to be a column-based architecture which infact represents the floor-planning of commercial architectures. Moreover a column-move operation is also implemented to optimize a column-based architecture for a set of netlists. Finally a much comprehensive set of netlists is used in this work.
Section 2 and 3 present a brief introduction of the coarse-grained exploration environment and the latest changes done in it. Section 4 gives the basic experimental methodology, the details of the benchmark circuits and the area model. In Section 5 the results and their analysis is presented.

## 2. Exploration Environment

The basic working ground of the exploration environment is a grid of equally sized SLOTS. CELLS of different sizes can be mapped on this grid. A CELL can be any block; a soft-block like a Configurable Logic Block (CLB); or a hard-block like an adder, multiplier or RAM etc. A CELL when mapped on the slot-grid is referred to as a SITE. Each SITE occupies one or more SLOTS. A routing channel passes between every two neighboring SITES. A SITE occupying more than one SLOT can allow routing channel to pass through it. A software flow maps the instances of a netlist on the SITES of its respective types. The PLACER, a software module, refines both the placement of SITES on the slot-grid (floor-planning) and the mapping of instances on the SITES (binding). The PLACER can also optimize an FPGA floor-planning for binding a set of netlists on it at mutually exclusive times. This technique of placing a set of netlists is proposed in [11], where it is used to explore configurable ASICs in a single dimension. This environment extends the same

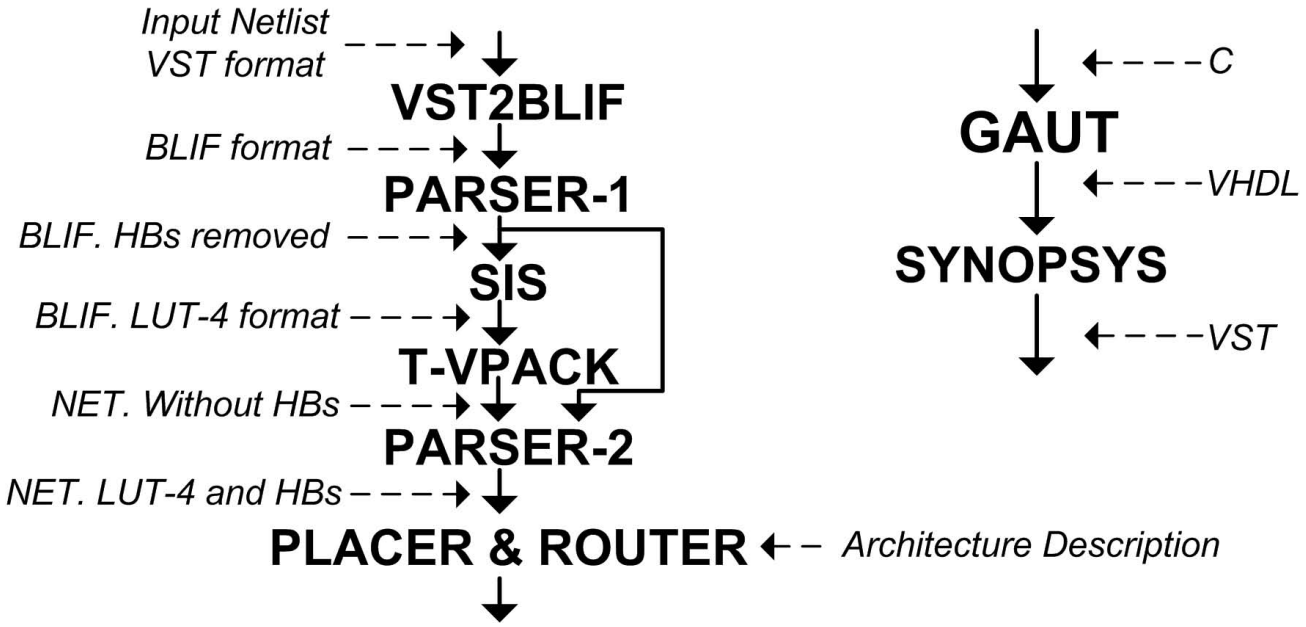



**Figure 2.1:** Software Flow

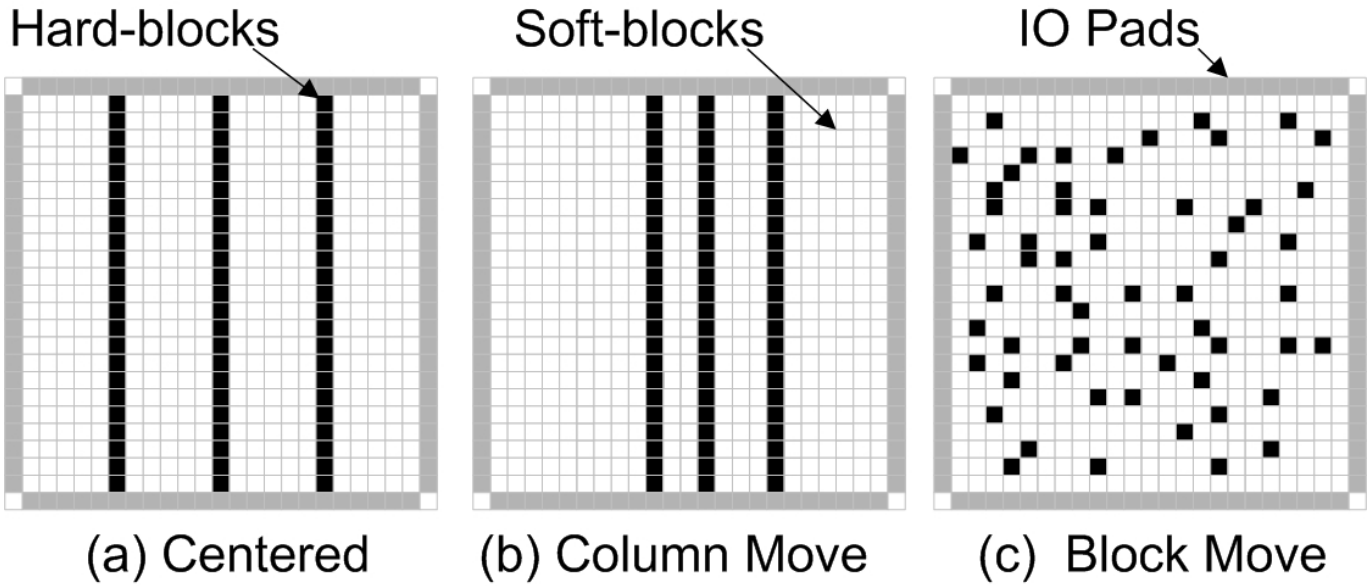



**Figure 2.2: FPGA Architecture Floor-plannings**

methodology to explore two dimensional island-style FPGA architectures. After floor-planning and binding, the ROUTER routes the netlist on the architecture.

**Software Flow:** An architecture description file is used to define the complete architecture description details. Once the architecture is properly defined, a software flow (as shown in Figure 2.1) maps the netlists on it. The input to this software flow is a structural netlist in VST (structured VHDL) format. This netlist can also be generated from 'C' code using GAUT [2]. The VST file is composed of the traditional standard cell library instances and the hard-block instances. VST2BLIF tool is modified to convert the VST file having hard-blocks to BLIF format. Later a PARSER removes all the instances of hard-blocks and passes the remaining netlist to SIS [17] for synthesis into 4-LUT format. All the dependence between the hard-blocks and the remaining netlist is preserved by adding new input and output pins to the main netlist. After SIS, and later by the conversion of the netlist to NET format through T-VPACK [9], another parser adds all the removed hard-blocks into the netlist. It also removes all the previously added inputs and outputs. This final netlist in NET format, containing LUTs and hard-blocks, is passed to the PLACER and the ROUTER. In future instead of SIS, we intend to use ABC [1].

## 3. The Placer

**Simulated Annealing Parameters:** The PLACER uses the Simulated Annealing algorithm [8] [15] to minimize the placement cost. In this work all the parameters of simulated annealing are set to be the same as that used in VPR [8]; i.e. the method of setting the initial temperature, the number of moves per temperature step, the method of updating the annealing temperature, the formula for accepting a move operation and the terminating condition of the annealer.
This work also applies the Range Limiter (RLimit) operation for coarse-grained architectures. It is shown in [8] [14] that it is desirable to keep the fraction of moves accepted near 0.44 for as long as possible. This is achieved through the Rlimit functionality which limits an interchange operation to be restricted to the rectangular space in its surrounding. The rectangular space diminishes with the decrease in acceptance ratio. Thus the more the interchange operation takes place in its vicinity, the more probable it is to be accepted and thus maintains the fraction of moves accepted. Generally in case of architectures containing homogeneous blocks, the Rlimit operation is implemented by randomly selecting the destination position from within the rectangular space surrounding the source. Whereas in case of heterogeneous architectures, the range limiter also needs to verify that the randomly selected destination SITE from within the range limit must be of the same type as the source SITE. Hence some additional execution time is required.

**Bounding box formation:** The bounding box (bbx) of a signal or a net is the minimum rectangular area that contains the driver instance and all the receiver instances of the net. The PLACER tries to minimize the sum of the half-perimeters of the bounding boxes of all the nets. Since the hard-blocks in a coarse-grained architecture can span to two or more slots. Thus for more precise placements costs the position and direction of pins are also considered in the formation of bounding box. Similarly all the input pins of a SITE having same class are also included in the bbx. Thus the definition of the bounding box used in this work is the minimum rectangular area that contains the driver pin and the receiver pins of the net, and all the input pins of a SITE having the same class as that of the receiver pin of the SITE connected to the net.

**Placer Operations:** Different kind of operations are implemented to optimize the placement costs. The PLACER either **(i)** moves an instance from one SITE to another, **(ii)** moves a SITE from one SLOT position to another, **(iii)** rotates a SITE at its own axis, or **(iv)** moves a complete column of SITES from one SLOT position to another. After each operation the placement cost is recomputed for all the disturbed nets. Depending on the cost value and the annealing temperature the operation is accepted or rejected. Multiple netlist files can be placed together to get a single architecture floor-planning for all the netlists. With multiple netlist placements, each SITE can allow multiple instances to be mapped on it; but multiple instances of the same netlist cannot be mapped on the same SITE.

The PLACER operation is randomly applied to an instance of any of the input netlists, or to a SITE in the architecture. If the selected SITE is restricted to remain in a column, then a SITE move operation will move the complete column along. A SITE which is not restricted to remain in a column cannot be swapped with a SITE in a column.
In case of multiple netlist placements, the placement operation is applied both on the instances of each of the netlists, and on the SITES in the architecture. In case of SITE movement, the change in the placement costs of all the instances mapped on the SITE influence the acceptance or rejection of the SITE move. The SITE movement acceptation is done according to some weight assigned to each netlist. This helps improve the floor-planning more in favor of some netlist than others. The experimentation performed in this work gives same weight to all the input netlist.

## 4. Experimentation

The main aim of this experimentation is to compare different aspects of a column-based layout with a non column-based layout of a coarse-grained FPGA. A variety of test benches are collected which are further distributed into groups (each group containing 3 to 5 netlists). The aim is to find an FPGA floor-planning for each group. All the netlists in a group influence the floor-planning thus resulting in a trade-off floor-planning. For each group three type of floor-planning are explored. (i) A fixed column based floor-planning in which the hard-blocks of each type are placed in columns at equal spacing (as shown in Figure 2.2 (a)) (ii) A trade-off floor-planning achieved through a column move operation (shown in Figure 2.2 (b)) (iii) A trade-off floor-planning achieved through a block move operation (Figure 2.2 (c)). Initially the floor-planning (ii) and

| Block Name | Netlists | | | | Target | Block Sizes | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| | **Fir** | **Fft** | **Adac** | **Dcu** | Target-1 (18x12) Supply Ratio 1:5 | Block Size Lamda$^2$ | Inputs | Outputs | Slots for ch >= 9 | Block Size/Shape | % free space for ch=11 |
| clb | 32 | 94 | 47 | 34 | 94 | 58500 | 4 | 2 | 1 | 1x1 | - |
| mul_8_8_16 | 4 | 4 | - | 1 | 4 | 1075250 | 16 | 16 | 9 | 3x3 | 1.56% |
| slansky_16 | 3 | 3 | - | 1 | 3 | 306750 | 32 | 16 | 8 | 2x4 | 2.62% |
| sff_8 | 4 | - | 2 | 4 | 4 | 36000 | 8 | 8 | 3 | 2x2 | 13.78% |
| sub_8 | - | 6 | - | 2 | 6 | 154500 | 17 | 8 | 4 | 2x2 | 1.64% |
| smux_16 | - | - | 1 | 2 | 2 | 36000 | 33 | 16 | 8 | 2x4 | 6.83% |
| | **Fir16** | **Prodmat** | **Ellipt** | | Target-2 (36x36) Supply Ratio 1:26 | Block Size Lamda$^2$ | Inputs | Outputs | Slots for ch >= 24 | Block Size/Shape | % free space for ch=23 |
| clb4 (4 clbs) | 572 | 1112 | 818 | - | 1112 | 798000 | 10 | 4 | 1 | 1x1 | - |
| add_16_16_16 | 8 | 11 | 15 | - | 15 | 106750 | 32 | 16 | 3 | 1x3 | 26% |
| mul_16_16_16 | 16 | 27 | - | - | 27 | 1908750 | 32 | 16 | 4 | 1x4 | 18% |
| | **Fft** | **Lms** | **Prodmat** | **Conv** | Target-3 (52x42) Supply Ratio 1:7 | Block Size Lamda$^2$ | Inputs | Outputs | Slots for ch >= 47 | Block Size/Shape | % free space for ch=65 |
| clb10 (10 clbs) | 1349 | 1310 | 1665 | 1077 | 1665 | 3675000 | 22 | 10 | 1 | 1x1 | - |
| add_18_18_18 | 180 | 34 | 64 | 58 | 180 | 241500 | 37 | 19 | 2 | 1x2 | 49.3% |
| mul_18_18_36 | 52 | 16 | 52 | 52 | 52 | 2498300 | 36 | 36 | 2 | 1x2 | 8.14% |

**Table 1: Netlist block utilisation table**

(iii) are attained for each group. Then each netlist in a group is individually mapped on each of the three floor-plannings. The difference in placement costs, routing channels required to route a netlist, and the total number of switches used for routing a netlist are compared for the three architectures. Finally the overall area of complete FPGA is compared using an area model.

### 4.1. Benchmarks

**Real Benchmark:** We have a first set of real netlists shown in Table 1. These netlists are divided into 3 groups. The first group comprise of 4 netlists which are a subset of 8-bit Fir, Fft, Adac and Dcu algorithms. These benches are extracted from their generator written in a procedural language. The Configurable Logic Block (CLB) used by these netlists is a simple 4 input Look-Up Table (LUT). Besides CLB, 5 different types of hard-blocks are used by these netlists. The second group of netlists contain 3 netlists comprising of Fir16, Prodmat and Ellipticass. These benches are generated from the 'C' algorithms using GAUT [2]. The CLB used by these netlists consists of 4 Basic Logic Elements (BLE), where each BLE is a 4 input LUT. Similarly the third group of netlists contains 4 netlists comprising FFT, LMS, Prodmat, and Convolution algorithms; These benches are also generated from the 'C' algorithms using GAUT [2]. The CLB used by these netlist consits of 10 Basic Logic Elements (BLE), where each BLE is a 4 input LUT. Each CLB containing multiple BLEs are connected through a full cross bar. The block nomenclature of each of these netlists can be seen in column 2-5 of Table 1. The target architecture sizes for each of these 3 groups of netlists, their supply ratios (Hard-Blocks : CLB), and their block requirements can be found in the column 6 of the table. The supply ratios are rounded up and are combined i.e (all type of hard-blocks : CLB). The remaining entries of the table will be discussed in the section "Area model".

**Synthetic Benchmark:** In order to give support to the above three groups of real netlists, some non-real (synthetic) testbenches are also used. The major aim of using these benches is that netlists of different demand-ratio containing a variety of blocks can be tested on architectures of different supply ratio. Since the main concern of this work is to find the effect of column based floor-planning on the placement cost, the routing channel requirement and eventually the area; this effect can be produced with any type of block, provided they are restricted to columns. For this purpose a group of 5 MCNC netlists (s298, apex2, seq, diffeq and misex3) have been selected. Some randomly chosen CLBs are renamed to type BLK1, BLK2 and BLK3 which are to be restricted to columns. A total of 12 different groups of netlists (each group comprising of the above 5 netlists) are used. The characteristics of these netlists can be found in the Table 2. The first 5 groups in the table contain only 1 type of block other than CLB. The next 4 groups of netlists contain 2 types of blocks other than CLB, and the last 3 groups contain 3 different blocks other than CLB. The target FPGA size of all these 12 groups is 44x44. The number of hard-blocks (in this case BLK1, BLK2 and BLK3) in each netlists are exactly multiple of 44; the demand ratio of each group changes with different number of CLBs in each netlist. In a 44x44 FPGA, a supply ratio of 1:43 means that the architecture contains 1 column containing 44 hard-blocks, the remaining 1892 are CLBs. 1:21 contains 2 columns, 1:14 contains 3, 1:10 contains 4 and 1:6 contains 7 columns. Since in a column based architecture a single column cannot be shared by 2 or more type of blocks. This is why a supply ratio of 1:43 is not tested with blk-2 and blk-3, and

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Group No. | Supply Ratio* | Demand Ratio** | Num of Blocks | Num of BLK1 | Num of BLK2 | Num of BLK3 |
| 1 | 1:43 | 1:38 | 1 | 44 | - | - |
| 2 | 1:21 | 1:19 | 1 | 88 | - | - |
| 3 | 1:14 | 1:12 | 1 | 132 | - | - |
| 4 | 1:10 | 1:9 | 1 | 176 | - | - |
| 5 | 1:6 | 1:5 | 1 | 308 | - | - |
| 6 | 1:21 | 1:19 | 2 | 44 | 44 | - |
| 7 | 1:14 | 1:12 | 2 | 88 | 44 | - |
| 8 | 1:10 | 1:9 | 2 | 88 | 88 | - |
| 9 | 1:6 | 1:5 | 2 | 176 | 132 | - |
| 10 | 1:14 | 1:12 | 3 | 44 | 44 | 44 |
| 11 | 1:10 | 1:9 | 3 | 88 | 44 | 44 |
| 12 | 1:6 | 1:5 | 3 | 132 | 88 | 88 |

* For FPGA Size of 44x44
** Average of 5 netlists

**Table 2: Synthetic Netlist Characteristics**

| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| No. | Netlist | Operation | Iterations | Time(s) |
| 1 | Dcu, Adac, Fir, Fft | Binding + Column Move | 47140 | 79,90 |
| 2 | Dcu, Adac, Fir, Fft | Binding + Block Move | 47140 | 36,30 |
| 3 | Dcu | Binding | 3389 | 1,31 |
| 4 | Adac | Binding | 3678 | 1,01 |
| 5 | Fir | Binding | 2775 | 1,53 |
| 6 | Fft | Binding | 13198 | 5,12 |
| 7 | Fir, Prodmat, Ellipticass | Binding + Column Move | 748231 | 2100 |
| 8 | Fir, Prodmat, Ellipticass | Binding + Block Move | 748231 | 1590 |
| 9 | Fir | Binding | 83162 | 83,4 |
| 10 | Prodmat | Binding | 174877 | 213 |
| 11 | Ellipticass | Binding | 93106 | 126 |
| 12 | Fft, Lms, Prodmat, Conv | Binding + Column Move | 2699561 | 52100 |
| 13 | Fft, Lms, Prodmat, Conv | Binding + Block Move | 2699561 | 11700 |
| 14 | Fft | Binding | 303305 | 1030 |
| 15 | Lms | Binding | 135146 | 917 |
| 16 | Prodmat | Binding | 216863 | 690 |
| 17 | Conv | Binding | 176136 | 598 |

**Table 3: Execution time table**

the supply ration of 1:21 with blk-3. Although an FPGA floorplanning can be made a non-square to get more columns. But it gives rise to an inefficient placement cost [8]. This inflexibility of having a desired supply ratio with a desired number of block types is one of the major drawbacks of the column based architectures.

### 4.2. Area Model

An area model is devised to determine (i) the area of a single slot in the slot-grid and (ii) the number of slots required by each hard-block. This model is derived from a tile-based FPGA Layout generation methodology [5]. The view of a single CLB slot in a tile-based layout can be seen in Figure 4.1. The area model considers each slot to be a single tile containing the block, the connection boxes (for connecting inputs and outputs with the routing channel), the routing channels on its top and right side, and the switch box of the top right corner. The area of a slot depends on (i) the area of a block (ii) the total number of inputs and outputs of the block and (iii) channel width. The basic sizes of the blocks (without connection boxes and routing channel) are measured in ALLIANCE [12] using a symbolic standard cell library 'SXLIB'. These sizes are shown in the 7th column of Table 1. The number of inputs and outputs of each block can be found in the column 8 and 9 of the table. The only missing parameter for calculating the size of a slot is the channel width which cannot be exactly determined before the routing phase. But atleast the threshold channel widths can be found at which the slot occupancy of hard-blocks change. So a threshold channel width of 9, 24 and 47 are found for Target-1,Target-2 and Target-3 architectures respectively. If the maximum routing channel required by these target architectures is below these thresholds then the slots required by the hard-blocks can be further reduced. The smallest block in the target architecture is made equivalent to be occupying 1 slot which is a CLB, CLB4 and CLB10 respectively in the three target architectures. Once the size of a single slot is determined, the sizes of all the remaining blocks are measured in units of number of slots occupied. Column 10 of Table 1 shows the slots required by each block for a threshold channel width. There is certainly some wastage of area when the blocks are represented in terms of number of slots occupied. Similarly while deciding the shapes of the blocks, the architect might require to further increase the area. The final sizes

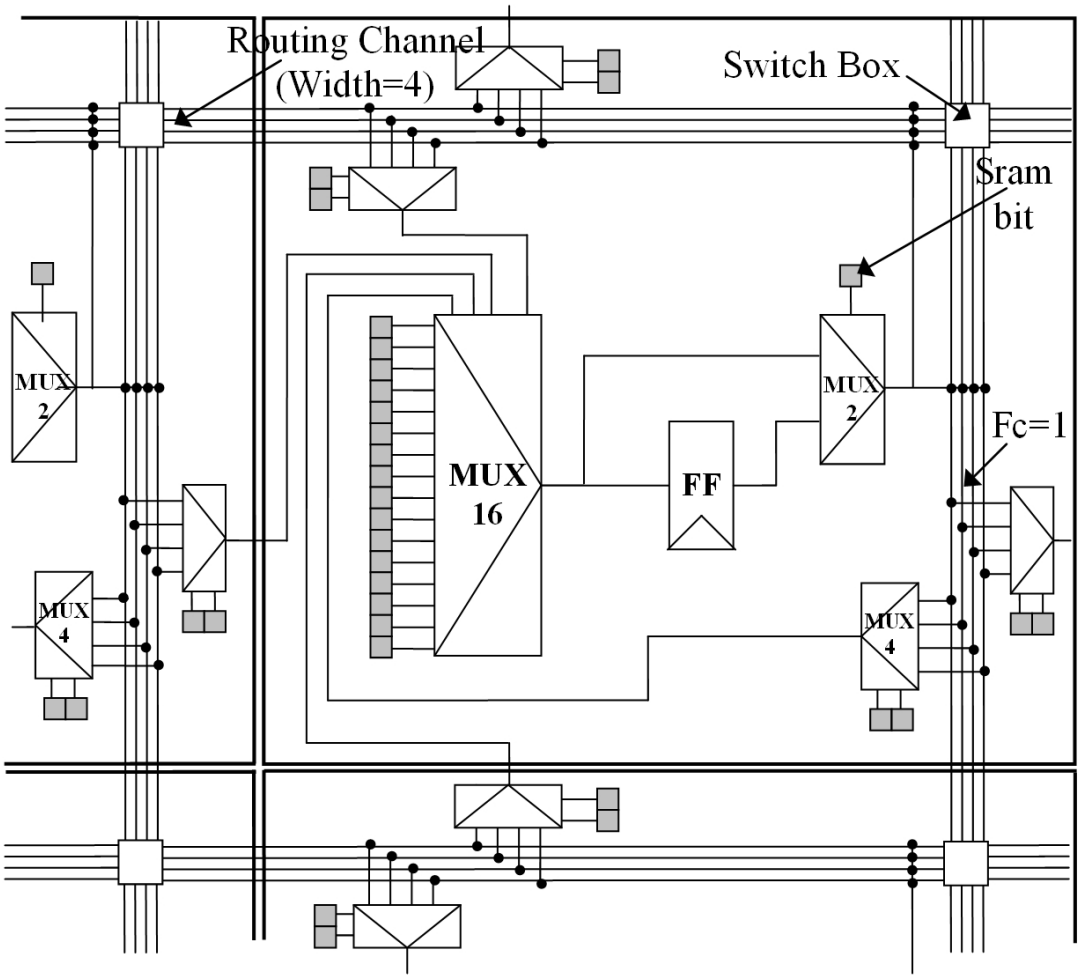
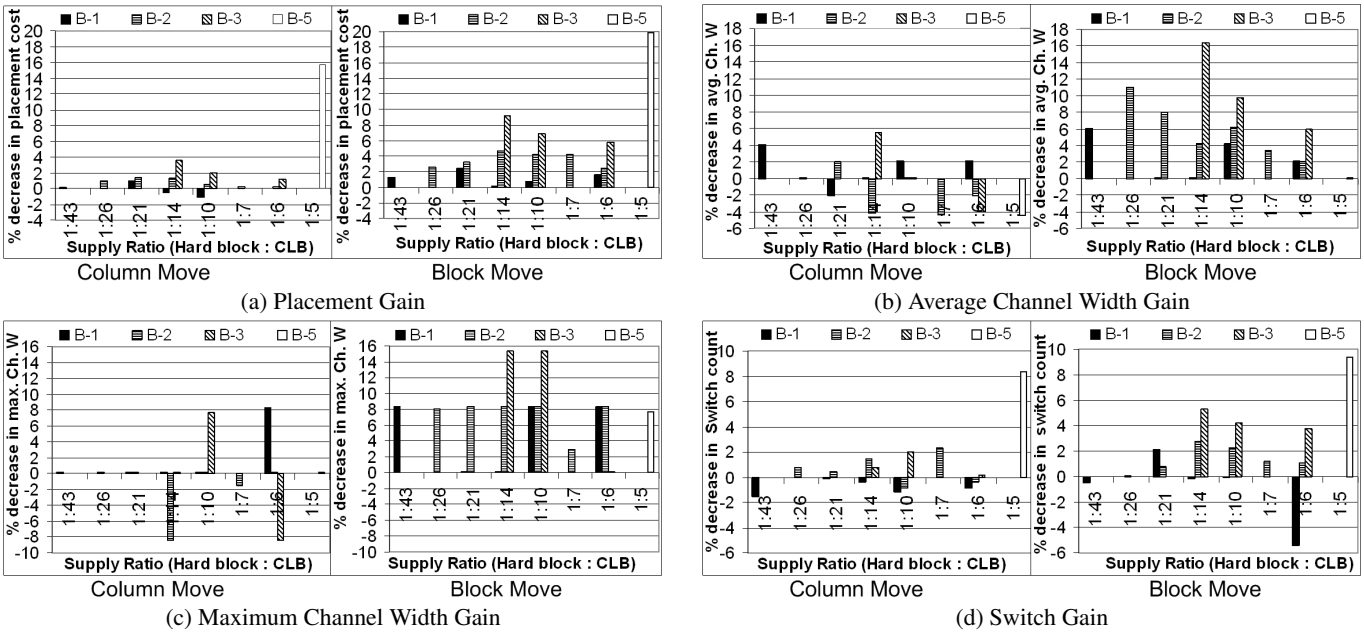



**Figure 4.1: View of a CLB Tile abutted with neighboring tiles**

and shapes of blocks are shown in column 11 of Table 1. The free area in each block is also reported by the area model. Column 12 of the table shows the percentage free area in each block. In this work no effort has been made to minimize the free area in each block. This free area can be used to integrate any auxiliary functionality. One of the uses of this free area can be to integrate shadow clusters [13] so that the precious routing resources can be reutilized if the hard-block is not used. The experimental analysis with shadow clusters is left for future work. Besides there can be many other architectural parameters in an FPGA (like the input and output connectivity of blocks (Fcin, Fcout), the CLB cluster size, etc) that are optimal if they are in a certain range. These parameters can always be changed to reduce the free area in each hard-block.

## 5. Results and Analysis

Three groups of real benchmarks and 12 groups of synthetic benchmarks are used to gather results. For each group, 3 kind of floor-plannings are generated; centered (Figure 2.2-a), trade-off floor-planning achieved through column move (Figure 2.2-b), and the trade-off floor-planning achieved through block move (Figure 2.2-c). All the netlists in a group influence the generation of column move and block move floor-planning. Table 3 shows the execution time of placement for the real benchmarks. The column 3 of the table shows the total number of inner iterations per temperature step. Column 4 of the table shows the total execution time of different placement operations on a 2GHz Intel Processor. The time to perform a Column Move or Block Move operation on a group of netlists is more than the sum of execution time for binding all the netlists of the group. Binding operation is also performed with the column move and the block move operation. The implementation of Column Move is still a bit un-optimized, so it takes more time than the Block Move operation. For the architectures containing different types of blocks, the columns in the centered architecture (Figure 2.2-a) are assigned to different blocks in such a way that each type of block




(a) Placement Gain

(b) Average Channel Width Gain

(c) Maximum Channel Width Gain

(d) Switch Gain

**Figure 4.2: Comparison results of column move and block move normalized to centered architecture**

should be near to other blocks. For example in case of group-11 in Table 2, the 4 columns of hard-blocks have been assigned to the blocks in the following order blk-1—blk-2—blk-3—blk-1. There can be a problem in assigning columns to hard-blocks for architectures having more than 3 type of hard-blocks. This is also a drawback of column-based architectures.

Once all the 3 floor-plannings are finalized, each netlist is individually placed (binded) on all the 3 floor-plannings. This ensures that the total number of inner iterations per temperature step are same for binding a netlist on all the three architectures. It can also be seen from Table 3 that as the total number of netlist in a group increase, the total execution time of block move or column move operation increases. This execution time can be reduced by parallelizing the move operation which is left for future work.

Figure 4.2 shows the percentage improvement of different results of column move and block move normalized to the centered architecture. For each group the placement gain (Figure 4.2-a), the average channel width gain of all the netlists in a group (Figure 4.2-b), the maximum channel width gain of any of the netlist in a group (Figure 4.2-c) and the total switch used gain (Figure 4.2-d) are shown. The vertical-axis represents the gain, whereas the horizontal-axis shows different architectures represented by their supply ratios. The architectures having supply ratios of 1:26, 1:7 and 1:5 represent the real benchmarks. The remaining architectures represent the synthetic groups. It can be seen in the figure that the architecture achieved through block move operation is generally much better than both the column move and the centered architecture. The placement gains of the block move operation results in better average channel width gain of all the netlists in each group. Similarly the maximum channel width required by any of the netlist in a group has also improved. And finally the switch gain of the architecture after block-move is also significant. The switch gain showed here is for the reduced channel widths. If the same channel widths as used by the centered architecture are used, the switch gains are much higher. It is also to be noted that column move has not shown as significant improvements as by the block-move operation. Instead in some cases the results have been deteriorated.

As the major portion of FPGA area is taken by the routing channel, a significant gain in routing area means a gain in the overall area. Generally the maximum channel width gain (Figure 4.2-c) represents the true gain. But in a column-based architecture, the column width of a hard-block can be easily adjusted according to the area needs. Whereas in a non-column based architecture, the height and width of each hard-block aught to be in multiples of a SLOT. So any waste in area of the hard-block will eventually negate the gains achieved in the maximum channel width. The column 12 of Table 1 shows the percentage area loss in each hard-block. So using the block sizes as shown in Table 1 and the maximum channel widths gain, we have calculated the total area gain of the block-move architecture compared to the centered architecture. In a column-based centered architecture we have not counted the free area of a hard-block. Whereas in a the block move architecture we have used the exact area shown in the Table 1 including the free area. We have noticed an area gain of 5% and 3.8% in the block-move Target-1 and Target-2 architectures. However in the Target-3

architecture we have a loss of 3.2%. The loss in Target-3 architecture is mainly due to the add_18_18_18 block which wastes nearly 50% of its area and has got 180 instances in the architecture. However if the free space in each hard-block is properly utilized by adding auxiliary functionalities, then the gains achieved can be more significant.

## 6. Conclusion and Future Work

We have presented a comparison of column-based coarse-grained FPGA architecture floor-planning with non-column based FPGA floor-planning. Producing the layout of the column-based architectures is relatively easier. Since only one kind of hard-blocks are found in a column, the widths of the column can be properly adjusted according to the area needs. However column-based FPGAs have relatively low placement results, which affects the routing channel requirement and the switches used for routing a netlist. The Column-based FPGA also puts a limit to the total number of hard-block columns in the architecture if a square FPGA architecture is required. A non-square FPGA can give more number of columns, but it produces more inefficient placements [8]. And lastly if a column of hard-blocks contains X instances of a hard-block, whereas the final application requirement is that of X+1 hard-blocks then it means that an extra column of hard-block is needed. Which in-turn means that there would always be some un-utilized hard-blocks.
On the other hand, a non-column based architecture layout optimized for a set of applications gives a much lesser placement cost, which eventually results in lesser routing channel requirement and lesser switch count (number of switches used in routing a netlist). The number of hard-blocks can be easily adjusted according to the needs while maintaining the overall FPGA architecture a square. However since both the height and width of hard-blocks need to be in multiples of CLB, there remains some free space in each hard-block. And the loss incurred by this unutilized free area might nullify or even further deteriorates the area in cases of having high supply ratio. However this unutilized free area in each hard-block can be used to integrate some auxiliary functionality in each hard-block. Another solution to minimize the area loss in each hard-block can be to represent the smallest block (a CLB) in multiple slots and not just 1 slot. In this way the height and width of hard-blocks need to be multiple of a smaller slot than the complete CLB. But this change requires more improvement in SITE movement techniques and even addition of some empty slots to fill in the gaps.
In future we intend to reduce the execution time of finding an architecture floor-planning for a set of netlists. This can be done by parallelizing the tasks. Finding an architecture floor-planning requires the movement of instances of netlists from one SITE to other, and the movement of SITES. So the movements of instances of different netlists can be done in parallel as they are not dependent on each other whereas the movement of SITE requires to be done sequentially with the movement of netlist instances.
In this work we have optimized only the floor-planning of an FPGA for a set of applications. This work can also be extended towards optimizing the reconfigurable routing channel for the set of input netlists.

## References

[1] Berkeley logic synthesis and verification group, ABC: A system for sequential synthesis and verification. *http://www.eecs.berkeley.edu/ alanmi/abc/*.
[2] GAUT - high-level synthesis tool from C to RTL, www-labsticc.univ-ubs.fr/www-gaut/.
[3] Stratix II device handbook. *Altera*, 2004.
[4] Virtex-4 family overview. *Xilinx*, 2005.
[5] .... Generic Techniques and CAD Tools for automated generation of FPGA Layout. *PRIME*, pages 141–144, 2008.
[6] .... A New Coarse-Grained FPGA Architecture Exploration Environment. *ICFPT*, pages 285–288, 2008.
[7] M. Beauchamp, S. Hauck, K. Underwood, and K. Hemmert. Embedded floating-point units in fpgas. *International Symposium on Field Programmable Gate Arrays (FPGA)*, pages 12–20, 2006.
[8] V. Betz, A. Marquardt, and J. Rose. *Architecture and CAD for Deep-Submicron FPGAs*. January 1999.
[9] V. Betz and J. Rose. VPR: A New Packing Placement and Routing Tool for FPGA research. *International Workshop on FPGA*, pages 213–22, 1997.
[10] C.H.Ho, P.H.W.Leong, W.Luk, S.Wilton, and S.Lopez-Buedo. Virtual embedded blocks: A methodology for evaluating embedded elements in fpgas. *Proceeding of FCMM*, pages 35–44, 2006.
[11] K. Compton and S. Hauck. Automatic design of area-efficient configurable asic cores. *IEEE Transaction on Computers*, pages 662–672, 2007.
[12] A. Greiner and F. Pecheux. Alliance: A complete set of CAD tools for teaching VLSI design. *3rd Eurochip Workshop*, 1992.

[13] P. Jamieson and J.Rose. Enhancing the area-efficiency of FPGAs with hard circuits using shadow clusters. *IEEE FPT*, pages 1–8, 2006.
[14] J.Lam and J.Delosome. Performance of a New Annealing Schedule. *DAC*, pages 306–311, 1988.
[15] Kirkpatrick, Gelatt, and Hecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.
[16] S.Dai and E.Bozorgzadeh. Cad tool for fpgas with embedded hard cores for design space exploration of future architectures. *FCCM*, pages 329–330, 2006.
[17] E. M. Sentovich and al. Sis: A system for sequential circuit analysis. *Tech. Report No. UCB/ERL M92/41, University of California, Berkeley*, 1992.
[18] C. Yu. A tool for exploring hybrid fpgas. *Proceeding of FPL PhD forum 2007*, pages 509–510, 2007.