AN EFFICIENT COMPREHENSIVE SURVEY ON ADVANCED ALGORITHMS FOR VLSI 3D PARTITIONING PHYSICAL DESIGN AUTOMATION

1*J SUNIL KUMAR, E. NAGARAJU2

12*DEPT OF ECE, ASSISTANT PROFESSOR, Vignan's Institute of Management and Technology for Women, GHATKESAR, TELANGANA 501301.

[1*]sunil5718@gmail.com [2]nagarajue83@gmail.com

A B S T R A C T

Today, VLSI technology has taken a fundamental role in developing most of the innovative electronic circuits. Despite the fact that VLSI design is eminent for its smaller size, lower cost, lower control, high reliability and high functionality, the design process takes long time and produces high risk. So, to get the knowledge with respect to the diverse contributions on VLSI design, the design of VLSI using streamlining is explored here. Accordingly, VLSI design advancement is analyzed through various algorithms and the execution measures of various VLSI experimentation are thought about. In 3D, the parceled units are assigned to a particular layer. So, we can say that layer assignment is a piece of 3D partitioning. Further, various improvements on 3D PARTITIONING PHYSICAL DESIGN Automation. Additionally, PARTITIONING problem of VLSI is considered and audited.

KEYWORDS:

VLSI design, Partitioning, K-L algorithm, Floor planning, Routing, Lee's algorithm, Layer Assignment, Hyper graph, 3D Vias, Wire-length, Max-cut.

I. INTRODUCTION:

The information revolution that has reworked our lives is driven by a revolution in integrated circuit (IC) technology. IC technology has evolved in the 1960s from the integration of a few transistors to the integration of millions of transistors in Very Large Scale Integration (VLSI) chips currently in use. The ascent in integration technology has been and continues to be created doable by the automation of varied steps concerned within the design and fabrication of VLSI chips.

The VLSI design cycle for the creation of a chip begins with a formal particular of a VLSI chip, follows a progression of steps, and eventually delivers a packaged chip. A typical design cycle might be spoken to by the flowchart in Figure 1. Our accentuation is on the physical design advance of this cycle.

ICs comprise of various electronic segments, built by layering several distinct materials in a well-characterized design on a silicon base called a wafer. The designer of an IC changes a circuit portrayal into a geometric depiction called the layout. A layout comprises of an arrangement of set of planar geometric shapes in several layers. The way toward changing over the particular of an electrical circuit into a layout is called the physical design process.

VLSI Physical Design (PD) Automation is essentially the examination, development and generation of algorithms and information structures related to physical design process.

The goal is to research optimal courses of action of gadgets on a plane (or in three measurements) and productive interconnection outline between these gadgets to fulfill certain topological, geometric, timing and power-utilization imperatives.

The PD procedure manipulates extremely simple geometric articles, for example, polygons and lines. As a result, PD algorithms have a tendency to be exceptionally instinctive in nature, and have noteworthy overlap with graph algorithms and combinatory improvement algorithms. In perspective of this perception, many consider PD mechanization the equivalent as the investigation of graph theoretic and combinatory algorithms for manipulation of items in two and three measurements The PD process itself is made of multiple sub-processes including: logic partitioning, floor planning and placement, routing (global routing and detailed routing), and compaction. This is illustrated in Figure 2. In what follows, we explain the problems involved in each of these steps and discuss the applications of a number of important optimization techniques, such as network flow, Steiner tree, scheduling, simulated annealing, generic algorithm, and linear/convex programming in solving them.

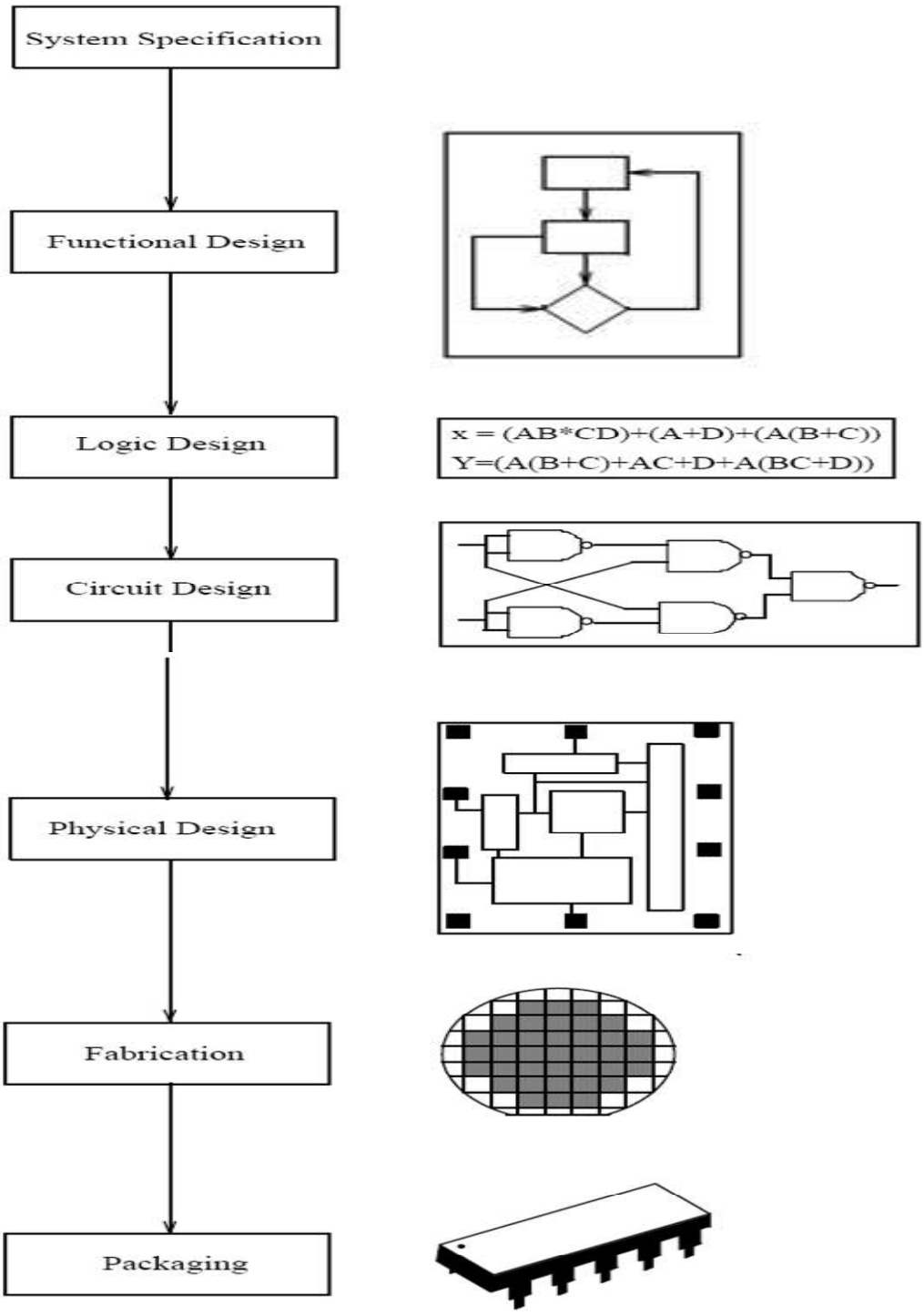


Figure 1: VLSI Design Cycle

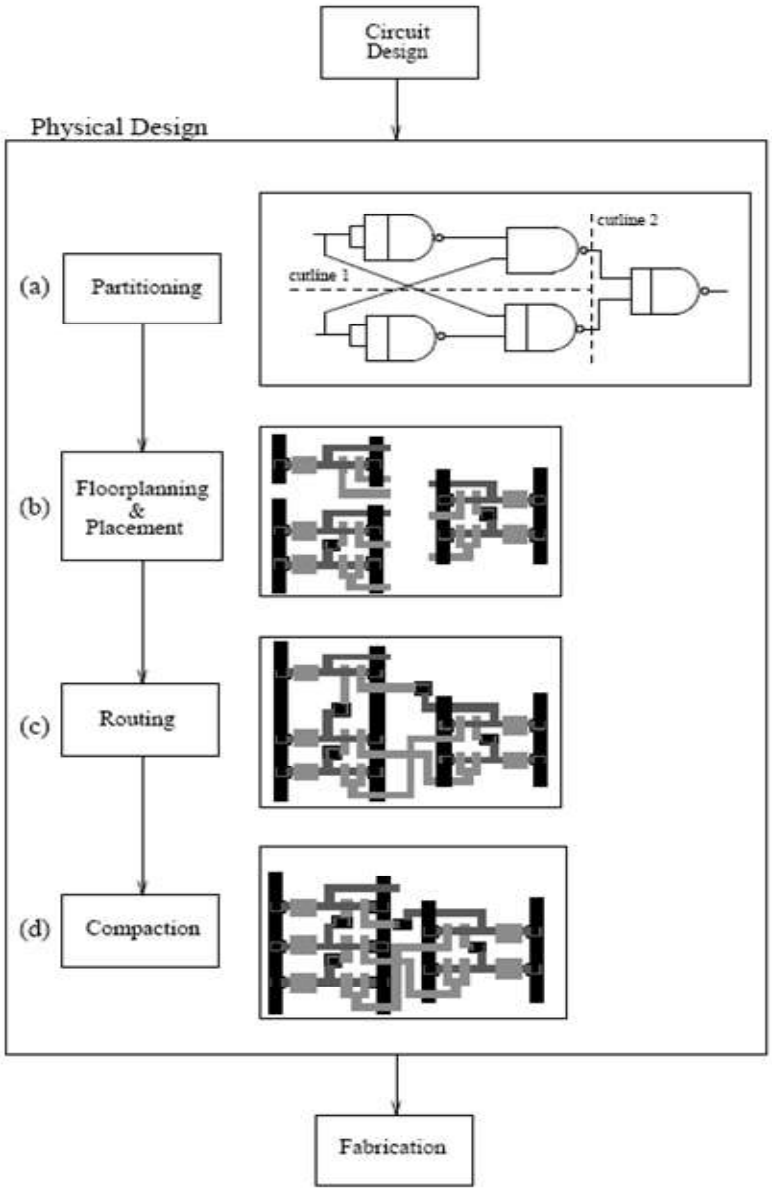


Figure 2: Physical Design Cycle

Classes of Graphs in Physical Design

The basic objects in Pd are rectangles and contours. The rectangles are used to represent circuit block tiles and empty tiles in a layout design. This is shown in Figure 3. The lines represent the interconnection wires. The relationship between these objects, such as overlap and distances, are extremely critical in development of PD algorithms. Graphs are a well developed tool accustomed study relationships between objects. Naturally, graphs are used to model numerous VLSI physical design problems and they play a pivotal role in almost all VLSI design algorithms

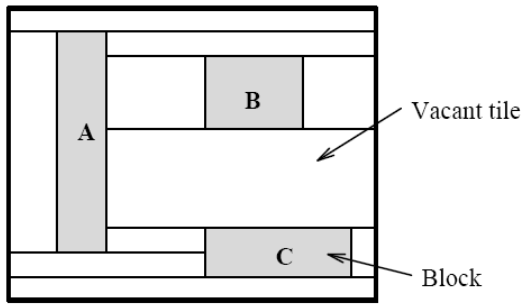


Figure 3: Layout partitioned into vacant and block tiles.

Numerous kinds of graphs are used to model PD problems. The three kinds of graphs related to a set of lines are (1) overlap graph: The graph has one hub for every line and an edge between two nodes if the corresponding lines overlap (without regulation) (2) control graph: There is an edge between two nodes in the event that one line completely contains the other and (3) interval graph: There is an edge between two nodes if there is an overlap or control between the corresponding lines. The graph used in relation with blocks is an area graph where two nodes (corresponding to two blocks) are interface if the corresponding blocks are contiguous.

Most of the enhancement problems in PD are NP-hard. Notwithstanding for the couple of problems that have polynomial time solutions, even quadratic time algorithms are sometimes infeasible because of the large number of components (> 106) involved. Another issue is the constants in the time complexity of the algorithms. In PD, the key thought is to develop practical algorithms, not just polynomial time complexity algorithms.

II. Partitioning

Partitioning is a procedure to separate a circuit or system into a collection of smaller parts (components). On one hand we can say, it is a design task to break a large system into pieces to be implemented on separate interfacing components and then again we can also say it serves as an algorithmic technique to solve difficult and complex combinatory improvement problems as in logic or layout synthesis. Partitioning has been a functioning zone of research for no less than a fourth of a century. The principle reason is that partitioning has turned into a central and sometimes critical design task today is the enormous increase of system complexity in the past and the normal further advances of microelectronic system design and manufacture. Synthesis and simulation tools regularly can't adapt to the complexity of the whole system a work in progress, and designers need to focus on critical parts of a system to speed-up the design cycle. Thus, the present state of design technology frequently requires a partitioning of the system.

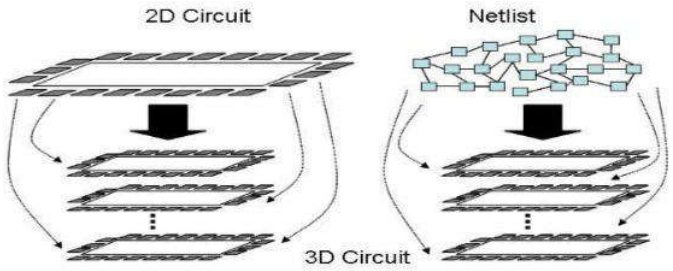


Fig 4. Migration from 2D to 3D VLSI integrated circuit.

While ongoing assembling technologies present many wire related issues because of process shrinking (such as signal respectability, control, delay), the 3D technology seems to significantly help the decrease of wire lengths [1, 2, 3] consequently lessening these problems. Once more, 3D technology also introduces its very own issues. One of them is the thermal dissipation problem, which is well studied at the floor planning level [4] as well as in placement level Another imperative issue acquainted is the manner by which with address the insertion of

the between level correspondence mechanism, i.e. 3D-Via, since it introduces important limitations to 3D VLSI design

The quantity of 3D-vias required in a design is dictated by the level assignment of every cell, which is performed amid the cell partitioning. The cell partitioning is performed by hyper graph partitioning tools such as hMetis [6] as done in [2]. Then again, hyper graph tools don't understand the distribution of partitions in the space (in 3D they are distributed in a single dimension) and fail to give optimal results. Partitioning applications exist on all levels of abstraction, specifically on the useful and therefore the structural (netlist) level. In the early stages of design, decisions must be made how to segment a design, frequently based on incomplete knowledge.

In particular, it has to be chosen whether to implement a segment in various types of equipment or software to accomplish an optimal size/execution exchange off because the granularity is low in this application. Since fully programmed partitioning is essential for fast iterations in design cycle, considerable exertion is made in the scholarly world and furthermore in industry to facilitate and enhance difficult decisions on functional level.

The figure 4 shows the possibility of 3D partitioning. Given a 2D placement netlist with pre-placed I/O pins at the limit of the locale available for cell placement, the relocation to a 3D netlist (prepared for 3D placement) has the following goals:

- Area allocation: the width and stature of the tiers must be calculated by the quantity of tiers.
- I/O partitioning: I/Os must be apportioned into various levels
- I/O placement: I/Os must be placed at the limit of the block, delimiting the territory for cell placement.

In this paper, next we discuss what number of types of partitioning there are. Next, we discuss about some effective, well known partitioning techniques. Next, finally we say about the future scope of this partitioning phase in 3D VLSI physical design.

TYPES OF PARTITIONING

Mathematically, partitioning problems are mostly formalized by using graphs. Functional descriptions can be modeled by signal flow graphs. The edges of this coordinated graph describe the signal flow between functional units which are represented by the graph's nodes. For modeling circuits as graphs on the structural level numerous alternatives have been published in the literature [14]. The modules of a circuit are often delineating as a collection of nodes. The signal nets interconnecting the modules can either be described by edges (coordinated or undirected) between pairs of nodes or by hyper edges interfacing sets of nodes (hyper graph). Edges or potentially nodes might be assigned weights, costs, or capacities. Hyper edges can be mapped into sets of (twofold) edges by using the model of a clique (complete sub-graph) or a star (by including additional nodes for the nets).

As a consequence of the varying applications various problem formulations for partitioning can be distinguished from an algorithmic perspective:

1. Two-way partitioning or bi-partitioning divides a graph into two non-void sub graphs while limiting the number or weight of the edges cut by the parcel. The aims to limit the quantity of slice edges and to get balanced partitions have been incorporated in the base proportion cut target work.

2. Multi-way partitioning divides a graph into a pre-specified number of sub graphs. The standard goal is to limit the quantity of edges between all partitions while

meeting constraints. Typical constraints are lower and upper size limits on the zone and the stick check of the components. Multi-way partitioning can be created by recursively solving two-way partitioning problems. For fast prototyping using arrays of FPGAs the multi-way partitioning problem is formulated as a decision problem [6].

3. Performance-driven partitioning is focused towards streamlining the system's execution as opposed to only limiting the quantity of interconnections between components. Most ongoing papers go for enhancing the planning [11, 12, 13], however minimal power consumption has also been used as a target.

4. Layout-driven partitioning emphasizes geometric aspects. The partitioning is based on a placement or geometric requesting as opposed to merely on structure (see [15, 16] and [9, 4, 2], e.g.). This is persuaded by the observation that a placement with limited wire length will lead to a low probability for a net being cut.

5. Partitioning with replication is useful to improve system execution, in particular in prototyping systems with FPGAs where the available FPGA pins are the limiting resource. By eliminating the necessity that the components of a parcel must be disjoint, the quantity of segment pins can be lessened and the system's execution enhanced by replicating parts of the netlist in various components.

ADVANTAGES OF 3D PARTITIONING

It is well known that 3D circuit technology has the potential of giving numerous improvements to VLSI circuits, including:

1. Largest wires decrease. It is clear that wire length can decrease going to 3D. First allow us to study the case of the largest circuit wire. The most extreme possible wire in a 3D chip equals to the half edge of the whole chip, which is width + stature. On 3D, both width and tallness will go down since dynamic region will be apportioned into at least two tiers. Das demonstrated this by experimental results [5].

2. Average wire length decrease. It is a conclusion of practically every paper on 3D circuit that the normal wire length goes down.

3. Dynamic Power decrease. It is natural to consider that diminishing overall wiring capacitance, power will go down.

4. Timing change. Benefits from timing are normal because of shorter connections.

5. Chip territory can be dramatically diminished with the expansion of dynamic tiers.

All the potential advantages must be explored with legitimate CAD tools that must have the capacity to characterize another design worldview solving new issues. The following section present a study of existing works on design effect of 3D circuits.

PARTITIONING ALGORITHMS

In this section an outline of alluring methods for solving partitioning problems will be given. Partitioning strategies are often classified as being constructive or repetitious. Constructive algorithms decide a partitioning from the graph describing the circuit or system, whereas iterative methods go for enhancing the quality of an existing partitioning solution. Partitioning algorithms also can be tagged settled or probabilistic Deterministic programs create the same solution each time they are started. Probabilistic methods result in contrasting solutions because they are based on arbitrary numbers. Constructive partitioning approaches are mainly based on

clustering, spectral or eigenvector methods, placement-based partitioning, mathematical programming, or network flow computations.

The initial partitioning-into - layers step is performed using the min-cut hMetis partitioning algorithm [Kim, C.; Shin, H. what's more, Yu, Y., 1995] and is additionally illustrated.

Using EV-Matrix

The goal of limiting the total vertical wire-length and most extreme cut between any two contiguous layers is equivalent to limiting the data transfer capacity of the EV-grid associated to the layer-blocks graph that resulted from the on top of partitioning [8]. The EV-network is an m×n grid where m – the quantity of rows – is the quantity of edges in the graph and n – the quantity of columns – is the quantity of nodes. An element a (i, j) =1 in the lattice is nonzero if the j-th hub is a terminal of the I-th net. In the event that a hub is not a terminal for a net, the corresponding EV-framework element is zero.

This procedure is briefly described in what follows using the graph example shown in Fig. 3. We first build the EV-network, and after that transform it into an as close as possible to a band-shape framework. This problem is meant as B (EV-lattice) - min problem. The method to solve this problem uses line and column flips so as to sort rows and columns such that non-zero elements are moved towards the principle diagonal. For example, for the grid shown in Fig. 3, keeping in mind the end goal to "float" non-zero elements from the upper half towards the fundamental diagonal column flips are performed between columns 2 and 3 and afterward column 6 is moved between columns 3 and 4. At the point when the above system finishes, the example from Fig. 3 becomes as final result. A lot of elaborated description is often found in [8]. The goal of getting the network to a band-frame serves two objectives:

Cut-size minimization – by having all 1's in the network clustered along the fundamental diagonal, the cut-size is limited wherever in the linear course of action

-WL minimization – by limiting most extreme distance spanned by any of the nets, the total wire - length of all nets is limited.

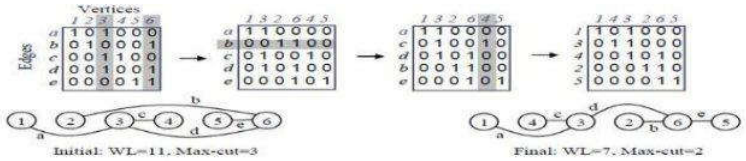


Fig.5. the EV-matrix to minimize the wire-length and max-cut between adjacent layers.

Using I/O Partitioning

One more paper [10] describes the decrease of 3D vias using I/O stick partitioning algorithm. For this methodology, we analyze the irregular logic block structure and make an I/O partitioning flow. He rule for I/O partitioning is delineating as follows:

1. Compute the logic distance.

2. Produce an entire graph of pairs of I/O pins considering the logical distance as a weight.

3. Perform the partitioning of complete graph going for min-cut streamlining and great number of pins between partitions.

4. Lock the I/O pins into partitions.

5. Perform the cells partitioning considering I/O positions.

6. Aspect proportion (from original netlist).

7. Pins introduction (from original netlist).

8. Whitespaces (from original netlist).

9. Legalize I/O positions.

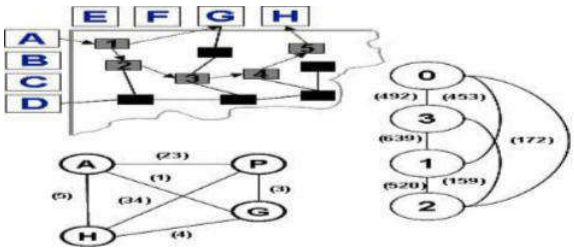


Fig.6. Illustration. Of shortest path between two I/O pins and a portion of correspondent.

The algorithm firstly calculates the logical distance between combine of I/O pins. Next, it creates a whole graph of I/O pins considering the logical distance as a weight. Finally, it partitions the graph victimization hMetis and considering the logical distance between I/O pins. The I/O pins are locked to its partitions. Based on the I/O pins location, the cells are divided. At last, the simulated annealing is applied to locate the best stacking course of action. The I/O placement preserves the same I/O pins introduction, whitespaces and aspect proportion of the original netlist. This strategy was named I/O pins. More details are often found in [7] and [8.

The values of shortest way are used to make a complete graph associating all pairs of I/O pins, as shown in Figure 4. For the cells partitioning, we tend to used hMetis tool [6]. The tool accepts weights for the cells. We assigned the inverse of the edge costs as their weights. We imposed a tight balance so as to keep a similar measure of me /Os in every level.

Using I/O Partitioning Refinement

In another paper [17] they present an iterative cells and I/O pins heuristic that refines the partitioning delivered by traditional hyper graph algorithms that further limit the 3D-Via tally while keeping up vertically shorter nets and keeping great I/O and cells zone balance. The algorithm picks an initial solution and improves it iteratively using irregular perturbations of the existing solution. The perturbations may be acknowledged or dismissed relying upon the cost variety. Any irritation that improves the present state is acknowledged and all perturbations that increase the cost are rejected.

The irritation work designed for our application attempts to move cells across partitions. Although they are irregular in nature, we perform two various types of perturbations for better diversity: single development or double trade. They work as follows:

•The single irritation can randomly pick a cell or an I/O pins (with half probability each) and move it to an alternate level (also chosen randomly).

• The double irritation randomly selects a couple of elements to switch partitions.

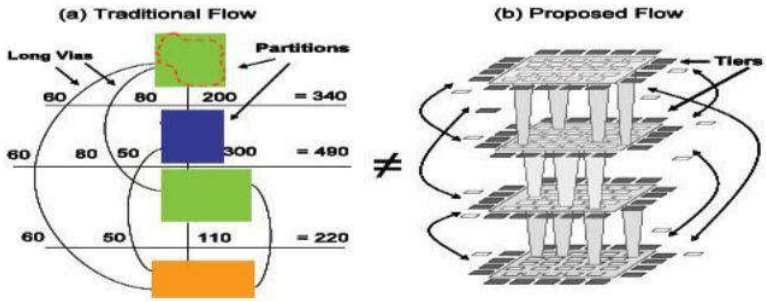


Fig.7. Fixed tiers method

The min cut problem is NP-complete. Hence, we use heuristic algorithms to solve the portioning problem in practice.

K-L algorithm: we have a tendency to currently discuss a renowned partitioning algorithm by Kernighan and sculptor United Nations agency printed it in their paper titled "An economical heuristic procedure for partitioning graphs" [2]. This rule (called the K-L algorithm) may be a bisectioning algorithm and divides the graph into 2 equal-sized partitions. (By applying it a number of time, we can get as many partitions as we want.) The objective is to minimize the cut size (because it is heuristic, it is not guaranteed).

The K-L algorithm is an iterative improvement algorithm. The algorithm starts with some initial partition. Local changes area unit applied to attenuate the cut size. Figure 5 shows an illustration of the K-L algorithm. The initial partitions are: A = {1,2,3,4} and B={5,6,7,8}. Note that the initial cutsize is 9. The next step of K-L rule is to settle on a try of vertices whose exchange ends up in the most important decrease of the cutsize or ends up in the littlest increase, if no decrease is possible. The amount by which the cut size decreases, if vertex vi changes over to the other partition, is represented by

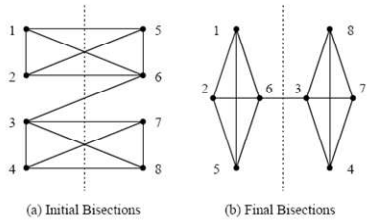
D(i)=inedge(i)- outedge(i)

Where inedge(i) (outedge(i)) is the number of edges of vertex i that do not (do) cross the bisection boundary. If two vertices vi and vj are exchanged, the decrease in cut size is D(i) + D(j). In the example, a suitable vertex pair is (3,5) which decreases the cut size by 3. A tentative exchange of this try is formed and therefore the 2 vertices area unit then secured. This lock on the vertices prohibits them from collaborating in any more tentative exchanges the on top of procedure is then applied to the new partition so on till all vertices area unit secured. During the process, a log of all tentative exchanges and the resulting cutsizes is stored. Figure (2) also shows this log table Note that the partial sum of the cut size decrease g(i) over the exchanges of first i vertex pairs is given in the table. The price of k that g(k) provides the most value of all g(i) is decided from the table. The first k pairs are actually exchanged. This completes an iteration and a new iteration starts. If no decrease in cut size is possible, the algorithm terminates.

The time complexity of the K-L algorithm is O(n3). This complexness is taken into account too high, even for moderately sized issues.. The K-L algorithm is, however, quite robust. It will accommodate further constraints, like a bunch of vertices requiring to be during a as partition.

```
Algorithm KL
begin
    INITIALIZE();
    while( IMPROVE(table) = TRUE ) do
    (* if an improvement has been made during last iteration,
    the process is carried out again. *)
        while ( UNLOCK(A) = TRUE ) do
        (* if there exists any unlocked vertex in A,
        more tentative exchanges are carried out. *)
            for ( each a ∈ A ) do
                if (a = unlocked) then
                    for( each b ∈ B ) do
                        if (b = unlocked) then
                            if (D_max < D(a) + D(b)) then
                                D_max = D(a) + D(b);
                                a_max = a;
                                b_max = b;
            TENT-EXCHGE(a_max, b_max);
            LOCK(a_max, b_max);
            LOG(table);
            D_max = −∞;
        ACTUAL-EXCHGE(table);
end.
```

The K-L algorithm.



i	Vertex Pair	$g(i)$	$\Sigma_{j=1}^{i} g(i)$	Cutsize
0	-	-	-	9
1	(3,5)	3	3	6
2	(4,6)	5	8	1
3	(1,7)	-6	2	7
4	(2,8)	-2	0	9

Illustration of the K-L algorithm.

III. Floor planning and Placement

Partitioning leads to blocks with well-characterized areas and shapes (settled blocks), blocks with inexact areas and no particular shape, a netlist specifying the connections between the blocks. The objectives of the floor planning and placement phase are to discover location of all blocks, the shapes of the flexible blocks and the stick locations for all the blocks.

Floor planning and placement can bigly affect the total length of the wiring used and the routing. This is illustrated in Figure .

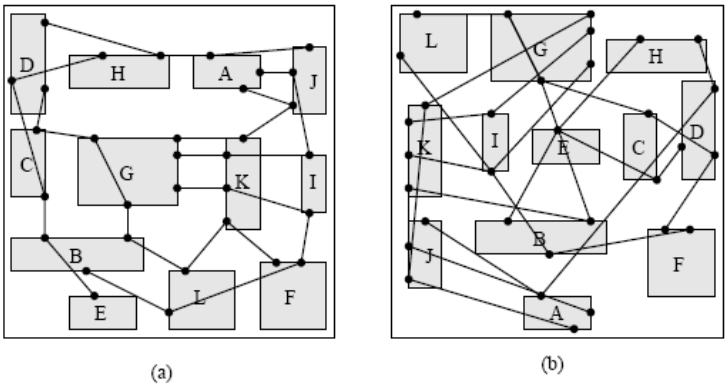


Figure: 8 Consequences of placement.

Given the inputs described above, the objective should satisfy the following restrictions. (1) No two rectangles overlap (2) Placement is routable (3) The total area of the rectangle bounding rectangles and net routing is minimized (4) The total wire length is minimized.

IV. Routing

In the placement section, the precise locations of circuit blocks and pins area unit determined. A netlist is additionally generated that specified the desired interconnections. Area not occupied by the blocks is often viewed as a set of regions. These regions area unit used for routing and area unit referred to as routing regions. Routing itself consists of 2 phases – international routing and careful routing. Figure eight illustrates the distinction between these 2 phases. Global routing generates a loose route for every web whereas careful routing finds the particular geometric layout of each net. We will only discuss Global routing to give a flavor of routing in general.

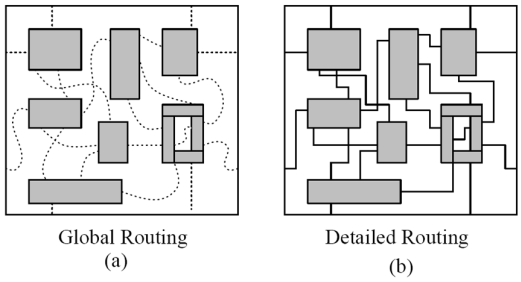


Figure:9 Routing.

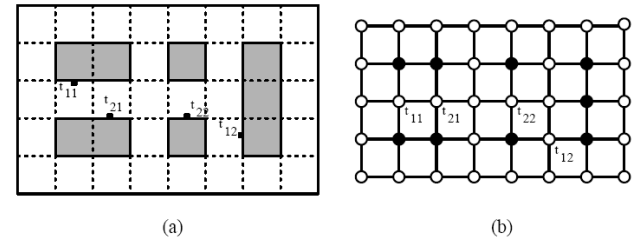


Figure: 10 Grid graph.

Framework graph is one of the graph models used for solving the routing problem. In this, every cell is represented by a vertex. Two vertices are joined by an edge on the off chance that they are neighboring one another. The involved cells are represented as filled circles whereas alternate cells are represented as clear circles. This is illustrated in Figure 9.

Global routing of multi-terminal nets may be developed as a Steiner tree downside. Given a netlist and the routing graph, discover a Steiner tree for each net.

Note that the matter of finding a Steiner is NP-complete. Consequently, we use rough algorithms to get an acceptable solution.

Lee's algorithm: This algorithm, which was developed by Lee in 1961, is the most widely used algorithm for finding a way between any two vertices on a planar rectangular lattice. The key to the popularity of Lee's labyrinth switch is its simplicity and its certification of finding an optimal solution in the event that one exists.

Lee's algorithm uses the Grid graph described previously. The exploration phase of Lee's algorithm is an enhanced version of the expansiveness first search. The search can be visualized as a wave spreading from the source. The source is labeled '0' and the wave front propagates to all the unblocked vertices neighboring the source. Each unblocked vertex contiguous the vertex labeled '0' is labeled '1' et cetera. This process continues until the objective vertex is come to or no further expansion is possible. The time and space complexity of Lee's algorithm is O(h*w) for a network of dimension h*w.

Different algorithms: Many different algorithms like Line-test algorithms, shortest way based algorithms and Steiner tree based algorithms exist for solving this problem. They involve distinctive approximations of solving the original Steiner tree problem.

After completion of elaborate routing, the layout is functionally complete. At this stage, the layout is prepared to be used to manufacture a chip. Be that as it may, due to non-optimality of placement and routing algorithms, some empty space is present in the layout. Keeping in mind the end goal to limit the cost, enhance execution and yield, layouts are decreased in size by expelling the empty space without altering the functionality of the circuit. This task of layout territory minimization is called layout compaction. Compaction is an exceptionally complex phase in physical design cycle. It requires understanding of numerous details of the creation process such as the design rules.

The layout of a VLSI circuit consists of geometric component (mostly of rectangular shape). The compaction problem can be stated as: Given a set of geometric features representing a layout and the base size of every segment and furthermore the base distance between every one of combine of components, the goal is to decrease the size of the components and to move the components towards one another while preserving the base sizes/distances.

CONCLUSION

This paper presented some proficient methodology for layer assignment i.e., apportioned components are assigned to various layers after initial partitioning so that the interconnection between the diverse layers can be limited. Besides we get better wire-length advancement results from this process. Since sequential hMetis is not all that proficient in case of wire-length minimization and max-cut streamlining problem. Here for all these algorithms we take final result of the multilevel hyper graph partitioning i.e., isolating the larger netlist as hyper graph into smaller sub-circuits using hMetis partitioning algorithm and relying upon the result we attract a graph such a route that after initial partitioning the no. of sub-circuits are assigned to various layers of equal size. In this paper, we investigated ways to additionally limit the wire-length. We study how to limit the wire-length and furthermore the quantity of 3D vias associated with it in the subsequent process. Because of increasing requirements on partitioning tools assist developments and improvements are extremely desirable. It has been observed in the past that small algorithmic modifications can be exceptionally compelling. From the application perspective, highly constrained execution driven partitioning is alluring for research and precise however productive delay calculation remains an essential issue. On the larger amounts of abstraction, applying logic synthesis methods seems to have awesome enhancement potential. The estimation of system properties needs consideration such that designers can analyze potential partitioning solutions quickly at the highest abstraction level possible. Last yet not least, synergy effects could result from organizing the partitioning activities on various levels of abstraction and for various applications at the real conferences.

REFERENCE

[1] C. Ababei, et al. “Placement and Routing in 3D Integrated Circuits”, IEEE Design and Test of Computers – special issue on 3D integration; pp 520-531, Nov.-Dec, 2005.

[2] B. Goplen, S. Sapatnekar, “Efficient Thermal Placement of Standard Cells in 3D ICs using Forced Directed Approach”, In: ICCAD’03, November, San Jose, California, USA, 2003.

[3] E. Wong, S. Lim, “3D Floorplanning with Thermal Vias”, In: DATE ’06, pp.878–883, 2006.

[4] S. Das, et al., “Technology, performance, and computer-aided design of three-dimensional integrated circuits”, In: ISPD’04, New York, NY, USA. ACM Press, pp.108–115, 2004.

[5] G. Karypis, et al “Hypergraph Partitioning: Application in VLSI domain”, In Proceedings of 34th Annual Conference on. Design Automation, DAC 1997, pages 526–529, 1997.

[6] G. Karypis, R. Aggarwal, V. Kumar, S. Shekhar, “Multi-level Hypergraph Partitioning: Applications in VLSI Design,” Proc. ACM/IEEE DAC, pp. 526-529, 1997.

[7] C. Ababei, K. Bazargan, “Non-contiguous Linear Placement for Reconfigurable Fabrics,” Proc. Reconfigurable Architectures Workshop (RAW), 2004.

[8] M. Motoyoshi, “Through-Silicon Via (TSV)”, In: Proceedings of the IEEE, Volume: 97,Issue: 1, On page(s): 43-48, 2009.

[9] S. Sawicki, “3D-Via Driven Partitioning for 3D VLSI Integrated Circuit”, CLEI ELECTRONIC JOURNAL, Vol. 13, No. 3, Paper 1, 2010.

[10] Y. Katsura, T. Koide, S. Wakabayashi, N. Yoshida, “A new system partitioning method under performance and physical constraints for multi-chip modules”, In Asia and South Pacific Design Automation Conference (ASP-DAC), pages 119–126. IFIP/ACM/IEEE, 1995.

[11] C. Kim, H. Shin, Y. Yu, “Performance-driven circuit partitioning for prototyping by using multiple FPGA chips”, In Asia and South Pacific Design Automation Conference (ASP-DAC), pages 113–118. IFIP/ACM/IEEE, 1995.

[12] R. Kuˇ znar, F. Brglez, “PROP: a recursive paradigm for area-efficient and performance oriented partitioning of large FPGA netlists”, In International Conference on Computer Aided Design (ICCAD), pages 644–649. IEEE/ACM, 1995.

[13] T. Lengauer, “Combinatorial algorithms for integrated circuit layout”. Wiley-Teubner, 1990.

[14] B. Riess, K. Doll, F. Johannes, “Partitioning very large circuits using analytical placement techniques”, In Design Automation Conference (DAC), pages 646– 651. ACM/IEEE, 1994.

[15] B. Riess, H. A. Giselbrecht, B. Wurth, “A new k-way partitioning approach for multiple types of FPGAs”. In Asia and South Pacific Design Automation Conference (ASP-DAC), pages 313–318. IFIP/ACM/IEEE, 1995.

[16] S. Sawicki, et al, “A Cells and I/O Pins Partitioning Refinement Algorithm for 3D VLSI Circuits”, IEEE Conference, pp. 852-855, 2009.