SANKEERNA: A LINEAR TIME, SYNTHESIS AND ROUTING AWARE, CONSTRUCTIVE VLSI PLACER TO ACHIEVE SYNERGISTIC DESIGN FLOW

Santeppa Kambham[1] and Siva Rama Krishna Prasad Kolli[2]
[1]ANURAG, DRDO, Kanchanbagh, Hyderabad-500058, INDIA
[2]ECE Dept, National Institute of Technology, Warangal-506004, INDIA

ABSTRACT

Standard cell placement is a NP complete open problem. The main objectives of a placement algorithm are to minimize chip area and the total wire length of all the nets. Due to interconnect dominance, Deep Sub Micron VLSI design flow does not converge leading to iterations between synthesis and layout steps. We present a new heuristic placement algorithm called Sankeerna, which tightly couples synthesis and routing and produces compact routable designs with minimum area and delay. We tested Sankeerna on several benchmarks using 0.13 micron, 8 metal layer, standard cell technology library. There is an average improvement of 46.2% in delay, 8.8% in area and 114.4% in wire length when compared to existing placement algorithms. In this paper, we described the design and implementation of Sankeerna algorithm and its performance is illustrated through a worked out example.

KEYWORDS**:** *Placement, VLSI Design flow, Synthesis, Routing, Area and delay minimization*

I. INTRODUCTION

VLSI chip complexity has been increasing as per the Moore's law, demanding more functionality, high performance, but with less design time. Producing compact layouts with high performance in shorter time is required, in order to meet the time to market needs of today's VLSI chips. This calls for tools, which run faster and also which converge without leading to design iterations. Placement is the major step in VLSI Design flow which decides the area and performance of the circuit. Detailed Routing is another time consuming step, which is performed after placement. If placement is not wire planned, routing may lead to heavy congestion resulting in several Design Rule Check (DRC) violations. It is required to iterate again with a new placement. If the wiring is not planned properly during placement, circuits routed may not meet the timing goals of the design. So there is a need for placers which are faster, produce compact layouts, meet the timing requirements and make the routing converge without DRC violations. The back end process of VLSI Design flow, that is, preparation of layout, is also to be tightly coupled with the front end synthesis process to avoid design iterations between synthesis and layout steps. It has been found that even after several iterations, this two step process does not converge and using wire load models this timing closure problem [1, 2] will not be solved.

In general, the standard cell placement problem can be stated as: Given a circuit consisting of technology mapped cells with fixed height and variable width, and a netlist connecting these cells, and Primary Inputs and Primary Outputs, construct a layout fixing the position of each cell without overlap with each other. The placement when routed should have minimum area, wire length, delay and should be routable. Minimum area is the area close to the sum of mapped standard cell areas. Minimum wire length is the sum of all nets in the circuit when placed and routed. Delay is the delay of worst path in the routed circuit. Routability indicates that the layout should not be congested; wires

routed should respect the Design Rules of the particular technology such that the routing is completed. Standard cell placement is known to be a NP complete open problem [3].
A Synergistic approach towards Deep Sub Micron (DSM) design, coupling logic synthesis and physical design is the need of the day [4, 1, 5]. There have been efforts to integrate synthesis and layout steps [6, 7, 8, 9]. All these efforts try to estimate wire delays, with the hope that they will be met finally, which is not happening. Wire delays are inevitable. The problem is not with wire delays, but with the non convergence and unpredictability. What we need is a quick way of knowing the final delay and a converging design flow. We have developed a design flow and a placer called *Sankeerna* targeted to produce compact routable layouts without using wire load models.
In Section 2, we briefly review the existing methods of placement and their limitations with respect to achieving a tightly coupled convergent design flow. Section 3 gives the basis for the *Sankeerna* algorithms. With this background, a new placer called *Sankeerna* was developed which is described in Section 4. The new placer *Sankeerna* is illustrated with an example in Section 5. The experimental setup to evaluate *Sankeerna* is described in Section 6. Results are tabulated and improvements obtained are discussed in Section 7. Conclusions of research work carried and future scope are given in Section 8.

II. RELATED WORK

Classical approaches to placement are reviewed in [10, 11, 3, 12, 13] and recent methods in [14, 15, 16]. The placement methods are classified based on the way the placement is constructed. Placements methods are either Constructive or Iterative [13]. In constructive method, once the components are placed, they will never be modified thereafter. An iterative method repeatedly modifies a feasible placement by changing the positions of one or more core cells and evaluates the result. Because of the complexity, the circuits are partitioned before placement. The constructive methods are (a) Partitioning-based which divide the circuit into two or more sub circuits [17] (b) Quadratic assignment which formulates the placement problem as a quadratic assignment problem [18, 19, 20] and (c) Cluster growth which places cells sequentially one by one in a partially completed layout using a criteria like number of cells connected to a already placed cell [21]. Main iterative methods are (a) Simulated annealing [22, 23,35], (b) Simulated evolution [15, 24] and (c) Force-directed [25, 20].
Another classification based on the placement technique used, was given in [20]. The placers were classified into three main categories namely (a) Stochastic placers which use simulated annealing which find global optimum with high CPU time, (b) Min-cut placers which recursively divide the netlist and chip area and (c). Analytical placers which define an analytical cost function and minimise it using numerical optimization methods. Some placers may use a combination of these techniques. These methods use only component cell dimensions and interconnection information and are not directly coupled to the synthesis.
The methods which use structural properties of the circuit are (a) Hierarchical placement [27] (b) Re-synthesis [9] and (c) Re-timing. There are algorithms which use signal flow and logic dependency during placement [28, 29]. In [28], critical paths are straightened after finding the zigzags. When placement is coupled with synthesis, this extra burden of finding criss-crosses is not required. In [30], using the structure of the interconnection graph, Placement is performed in a spiral topology around the centre of the cell array driven by a Depth First Search (DFS) on the interconnection graph. The algorithm has linear time complexity to the number of cells in the circuit.
To obtain delay optimized placements, timing driven placement methods are used [31, 32, 33, 34, 36, 37, 38, 39, 40]. The idea is to reduce the wire length on certain paths instead of total wire length. These methods are either path based or net based. The longest path delays are minimized in path based methods. Finding longest paths exponentially grows with the complexity of the design. Timing constraints are transformed into net-length constraints in the net based algorithms. Then a weighted wire length minimized placement is done iteratively until better timing is achieved. The drawbacks of this method are (a) delay budgeting is done without physical placement feasibility and (b) it is iterative. At the end of the iteration, the solution produced is evaluated.

To control congestion and to achieve routability, white spaces are allocated at the time of placement [41, 26, 42, 43]. The problem with this approach is, it increases area which in turn will increase wire length and delay. In [44], it was shown that minimising wire length improves routability and layout quality. Allocating white space may not be the right approach to achieve routability. It is better to minimise the wire length instead of allocating white spaces. The white space allocated may not be the right place required for the router.
The studies in [45] have shown that existing placement algorithms produce significantly inferior results when compared with the estimated optimal solutions. The studies in [12] show that the results of leading placement tools from both industry and academia may be up to 50% to 150% away from optimal in total wire length.
The design flow convergence is another main requirement of Synergistic approach towards DSM design [4]. Placement plays a major role in this. As mentioned in [4], there are three types of design flows for Deep Sub Micron (DSM) namely (a) Logic synthesis drives DSM design (b) Physical design drives DSM design (c) Synergistic approach towards DSM design. In the last method (c), it is required to create iteration loops which tightly couple various levels of design flow. The unpredictability of area, delay, and routability of the circuits from synthesis to layout, is another major problem. The study in [46, 47] indicated that the non-convergence of design process is due to non-coupling of synthesis [48, 49, 50, 51] to placement process. We need a faster way of estimating area and delay from pre-place to post-route. If we fail to achieve this, we may not have a clue to converge.
From the above analysis of the state of the art placement algorithms, we feel that there is still scope for improvement and the need for better placement algorithms meeting the requirements as mentioned in section 3. In the next Section, we describe the basis for our new algorithms, which try to solve some of the problems mentioned above.

III. BASIS FOR THE NEW ALGORITHMS

The new placement algorithm should have the following features (a) linear or polynomial time complexity with respect to number of cells in the circuit, (b) awareness of synthesis and routing assumptions and expectations, that is, tight coupling of synthesis and routing as mentioned in [4, 46, 47], (c) achieving minimum area and delay, (d) produce routable layouts without Design Rule Check (DRC) violations, by proper wire planning during placement, (e) delay of final layout should be predictable with trial routes and (f) should smoothly interface with synthesis and routing tools.
In this section, we explain the basis for the *Sankeerna* algorithms. Since the circuits are to be placed to achieve minimum area and delay, we first try to find out what is the minimum area and delay which are achievable for a given circuit and technology library. The minimum area achievable is the sum of widths of all cells multiplied by the height of the standard cells. For a given standard cell technology library, the height is same for all cells.
To find out the minimum delay achievable, we use Figure 1(a) to explain the delay calculation process.

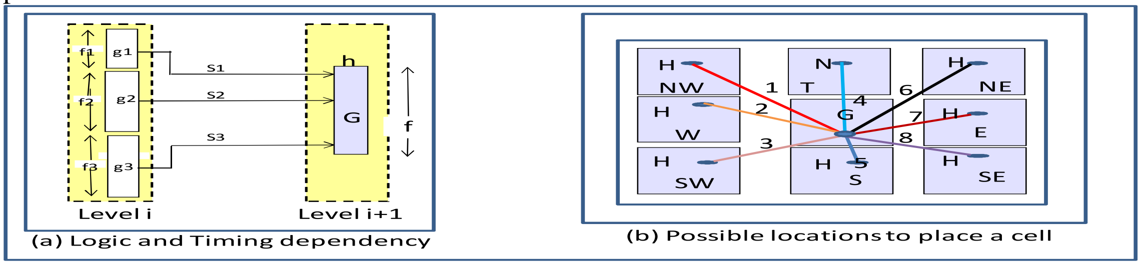

Figure 1 (a) Logic & timing dependency (b) possible locations to place a cell

There are 4 gates marked as g1, g2, g3 and G. The gates g1, g2 and g3 are at level i and G is at level i+1 as per the logic of the circuit. Let *d* is delay of gate G, *a* is maximum of arrival times of (g1, g2, g3) at the input of G, *b* is block delay of G, *f* is fan-out delay of G and *w* is wire delay of G. Then *d* is equal to sum of *a*, *b*, *f* and *w*. The *d* is dependent on the arrival times of inputs g1, g2 and g3 which

are in transitive fan-in of G. The gates g1, g2 and g3 in turn will have inputs either from Primary Inputs (PIs) or from other gates. So delay of G is dependent on the transitive fan in of all gates connected to g1, g2 and g3. To minimise *d*, the only thing that can be done at placement time is to minimize the wire delay *w*. Wire delay *w* depends on the wire lengths s1, s2 and s3, and several other factors. We will consider only wire length here. To minimize wire lengths s1, s2 and s3 which are outputs of gates g1, g2 and g3, they are to be placed at physically closest location to G. The possible places for a cell H which are nearer to a cell G are shown in Figure 1(b). H can be placed in any of the eight positions indicated by NW, N, NE, W, E, SW, S and SE. The distance from H output pin to G input pin for all these 8 possible locations depends on the width and height of H, G, and pin locations on these two cells. The lines 1, 2, 3, 4, 5, 6, 7 and 8 in Figure 1 show the Euclidean distance for all 8 positions. The same procedure can be adopted to calculate the Manhattan distance. The Physically Shortest Place (PSP) is the location which has minimum Manhattan distance. In real technology libraries, the cell positions are specified by a rectangle or a set of rectangles, not just as a point. So, we can choose to connect anywhere on the rectangle based on the closeness to the destination cell. Out of available locations, the one with the minimum Manhattan distance is on the Physically Shortest Path (PSP).
Let *r* be the required time of a gate, and *a* be the arrival time, then the slack *s* at gate G is *r* minus *a*. Out of all slacks of inputs to a gate G, the Worst Negative Slack (WNS) indicates that the cell is on the critical path. The inputs g1, g2 and g3 which are more critical are to be placed closer to gate G when compared to others. This argument has to be recursively applied to all gates which are in transitive fan in of g1, g2 and g3. That is, placing g1 nearer to G means, placing all the cells which are in transitive fan in of g1 nearer to G. All gates which are in transitive fan in of g1 are to be placed on PSP giving priority to cells which have higher WNS. Let WNS of g1, g2 and g3 be -2, -1 and -3 respectively. Placement priority is g3 first, then g1 and last g2. The minimum delay is achieved when g1, g2 and g3 are placed in PSP from Primary Outputs (POs) to Primary Inputs (PIs).
Sankeerna uses constructive method of placement. Starting from the Primary Output (PO), cells are placed on PSP as explained above. The height to width ratio of the smallest drive capability inverter is 4 for the standard cell library we have used for conducting experiments in this paper. So the row of a standard cell becomes the Physically Shortest Path (PSP). WNS at each node input, decides Delay-wise Shortest Path (DSP). *Sankeerna* combines PSP and DSP concepts explained above to produce routable placements minimizing area, delay and wire length in linear time. These concepts are further illustrated with a worked out example in Section 5. The next Section explains the algorithms used in *Sankeerna*.

IV. ALGORITHMS USED IN SANKEERNA

We have modified the algorithms of ANUPLACE [46] to produce delay optimized placements using a constructive method. *Sankeerna* reads the benchmark circuit which is in the form of a netlist, taken from “SIS” synthesizer [52], builds trees with Primary Outputs (POs) as roots. In SIS, we specify zero as the required time at the POs and print the worst slack at each node. This slack information is read along with the netlist into *Sankeerna*. The inputs are sorted based on this slack, with descending order of time criticality at each node. Starting from the root to the leaf node, nodes are placed on the layout after finding the closest free location. At each node, most time critical node is placed first using a modified Depth First Search (DFS) method. Priority is given to time when compared to depth of the tree. It was proved that placement of trees can be done in polynomial time [7]. A Depth First Search (DFS) algorithm was used in [30] which has linear time complexity to the number of connections. Tree based placement algorithms reported in literature have either linear time or O (n log n) time complexity [7, 53, 54, 55].
We have used benchmark circuits from SIS [52] synthesizer in “BLIF” format which are then converted into Bookshelf [56] format using converters provided in [59, 60, 41, 61]. The normal placement benchmark circuits [57, 45] are not useful because they give only cell dimensions and interconnect information. Timing, cell mapping, logic dependency and other circuit information from synthesizer are not available in these placement benchmarks. These converters do not use technology library information for cell dimensions and pin locations. *Sankeerna* reads the technology

information consisting (a) cell names, (b) cell dimensions height and width, (c) pin locations on the cells, (d) timing information, and (e) input load from a file. Using this technology information, *Sankeerna* generates the benchmark circuit in bookshelf format with actual cell dimensions and pin locations. Once the trees are created the slack information is read into the tree data structure. *Sankeerna* algorithm is shown in Figure 2.

```
Main
•Read technology library
•Read the benchmark circuit.
•Build trees with primary outputs as roots.
•Read cell mapping and delay file
•Print verilog
•Sort inputs of each node of the tree based on time criticality.
•Sort Trees of Primary Outputs (PO) based on time criticality.
•Put level information in each node of the tree.
•Print the benchmark with technology cell dimensions and pin locations.
•Run Public Domain Placer (PDP)
•Read PDP placement
•Calculate the layout width and height using standard cell area & aspect ratio.
•Number of rows = layout height / standard cell height.
•Initialize row tables to keep track of the placement as it is constructed.
•Place circuit .
•Place Primary Inputs (PIs) and Primary Outputs (POs)
•Print ".def" files of PDP and Sankeerna..
```

```
void place_ckt ()
{   next_PO = pointer to list of trees
    pointed by Primary Outputs(POs);
  no=number of  PO cells;
  for ( i=0; i<no; i++ )
   { place_cell ( next_PO );
     next_PO=next_PO->next;
   }
}
```

```
void place_cell ( gate )
{ next_pin = pointer to
    list of input pins;
  place_one_cell ( gate);
  for ( i=0; i< ( gate->no of inputs ); i++ )
   { place_cell ( next_pin );
     next_pin=next_pin->next;
   }
}
```

```
void find_best_place ( gate)
{ checkavailability on the same row,  and
    above and    below current row;
  Out of available rows, find the row
    which gives minimum wire length;
  return coordinates of minimum location;
}
```

```
void checkavailability_on_row (row, width)
{ x1= row_table_x1[row]; x2=row_table_x2[row];
  if (( fabs ( x2-x1 ) +width ) <=  layout_width )
     return(possible,x2)
  else return(not_possible);
}
```

```
void place_one_cell ( gate )
{ find_best_place (gate);
  place_cell_on_layout_surface ( gate);
}
```

Figure 2 *Sankeerna* algorithms

As shown in the Figure 2, the "place_ckt" function places the trees pointed by Primary Outputs (POs) one after the other using "place_cell" function starting at row 1 at x=0 and y=0. The "place_cell" function works as follows.

- Place the cell pointed by root using "place_one_cell".
- For each input, if it is a primary input, place it using "place_one_cell", if not; call "place_cell" with this input recursively.

The "place_one_cell" function finds the best place to place the cell using "find_best_place" function and places the cell at this location. The "find_best_place" function works as follows. As the placement progresses, the "used space" and "available space" are marked, C is the current cell and the next cell will be placed closer to this cell wherever space is available. The current cell placement is depicted in Figure 3.

The "find_best_place" function checks the availability of space using "check_availability_on_row" function on the same row of C and on the rows above and below the current row of C. The possible places for a cell H which are nearer to a cell G are shown in Figure 3. Out of available locations, the one with the minimum distance from the parent cell is chosen. The cell is placed at this location. The "check_availability_on_row" function keeps two pointers x1 and x2 for each row. Initially x1 and x2 are initialised to zero before the start of placement. When this function is invoked, it gets these two pointers x1 and x2 from the table corresponding to this row. It then calculates whether in the available space, this cell of "width" can be placed such that the row width will be less than or equal to the layout width. If space is available, it returns x2 as the available location on this row.

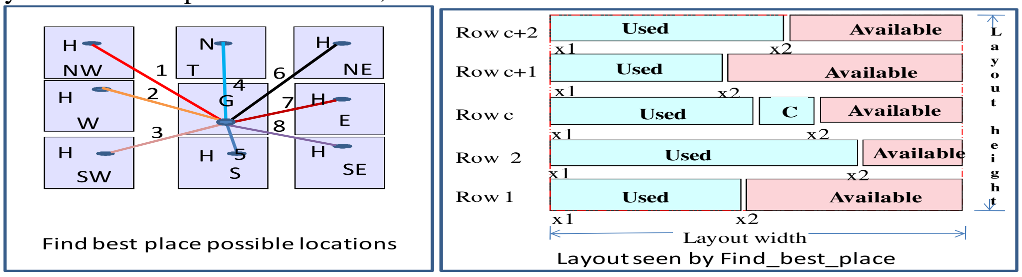

Figure 3 Find_best_place possible locations and the layout seen by Find_best_place

Complexity of the algorithm

If n is the number of the cells in the circuit, "place_one_cell" function is invoked n times. "Place_one_cell" calls "find_best_Place" which uses "check_availability_on_row". So "find_best_Place" is executed n times. Each time, it calculates wire lengths for possible locations to choose the best one. "check_availability_on_row" performs one comparison operation. So number of operations is linearly proportional to n, after construction of the tree. So the complexity of algorithm is of the order of n.

V. ILLUSTRATION OF SANKEERNA WITH AN EXAMPLE

The algorithms used in *Sankeerna* are illustrated with an example whose logic equation is given below, taken from [49].

$$Y = abe\overline{g} + abfg + ab\overline{e}g + ace\overline{g} + acfg + ac\overline{e}g + de\overline{g} + dfg + d\overline{e}g + bh + bi + ch + ci$$

The logic diagram with technology mapped cells and the tree built by *Sankeerna* for the above logic equation with the slacks are shown in Figure 4 and 5 along with the sequence of placement based on the time criticality after synthesis using SIS [52].

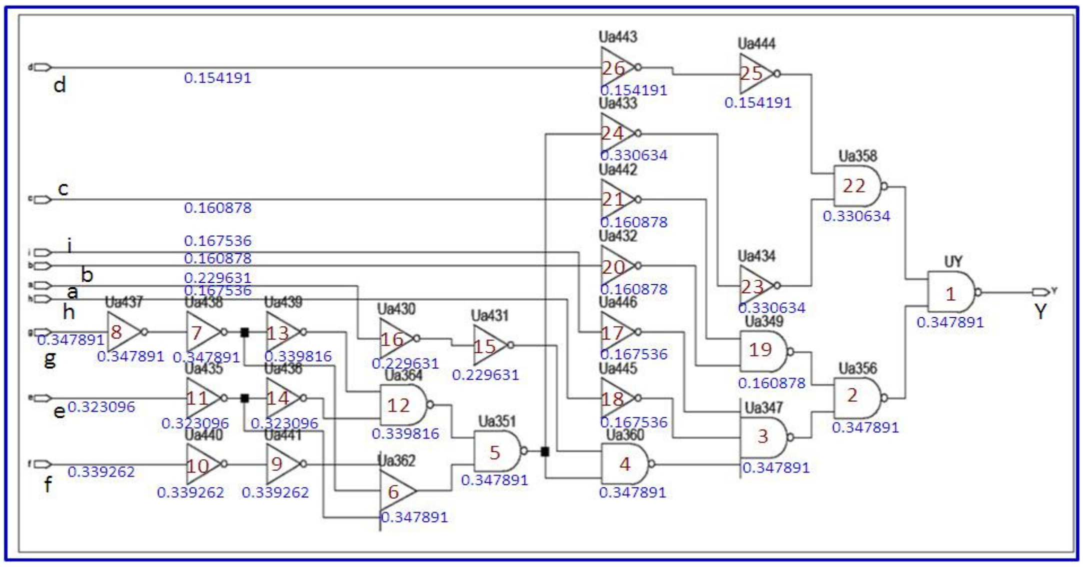

Figure 4 Logic Diagram with technology mapped cells for the example equation

The sequence of placement is indicated by the numbers 1-26 shown at the each node of the tree. There are 9 primary inputs marked as a, b, c, d, e, f, g, h, i and there is one primary output marked as Y. *Sankeerna* places the Primary Output cell Y first at x=0 in row 1. Then it looks at its leaf cells Ua356 and Ua358. From the time criticality given in Figure 5, it places cell Ua356 after finding the best place. The algorithm is then recursively invoked to place the tree with root as Ua356 which places the cells and the inputs in the sequence 3 to 21.

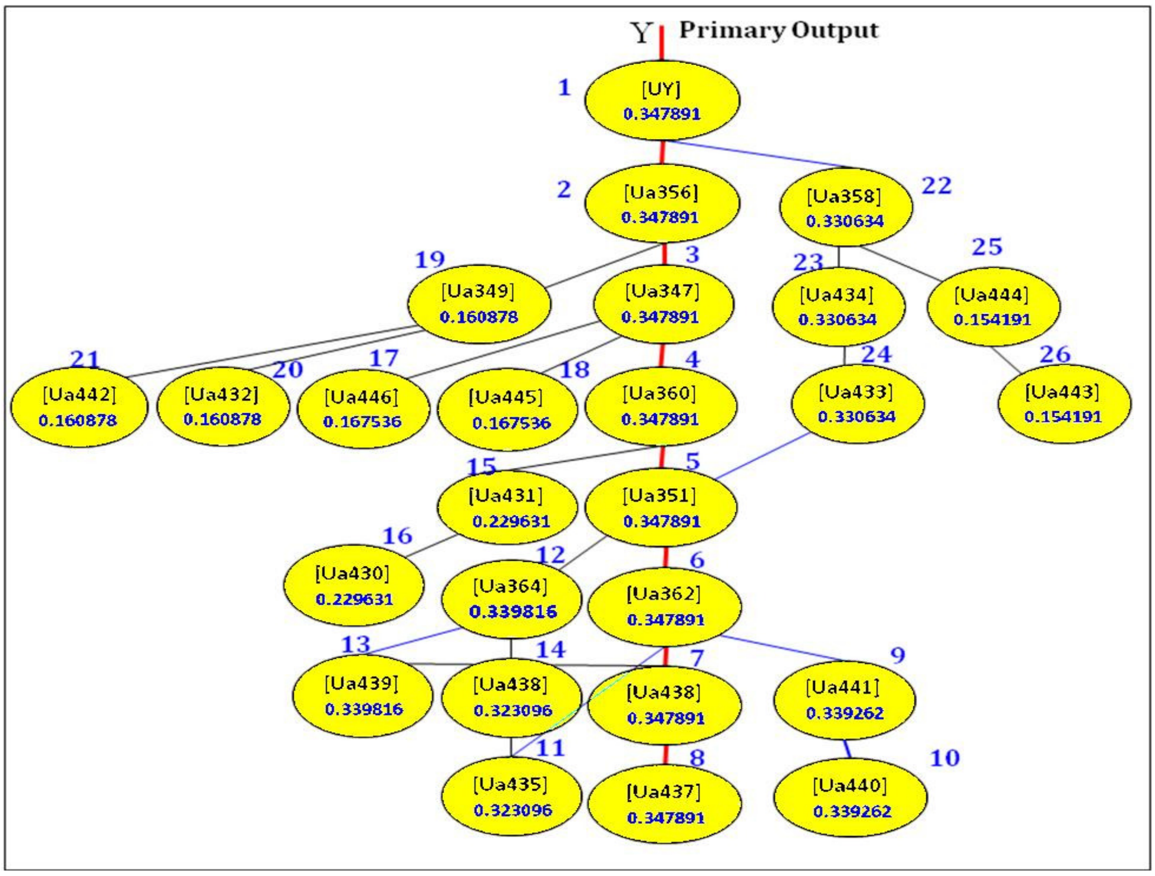

Figure 5 Tree built by *Sankeerna* for the example

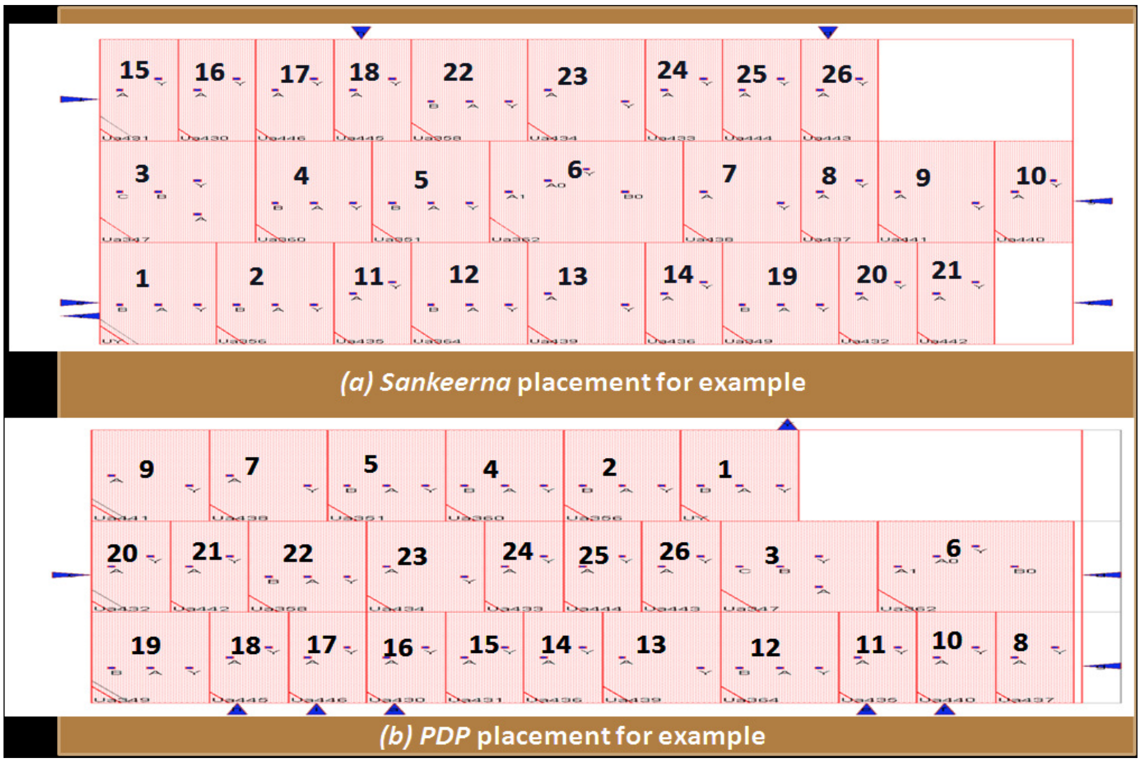
(a) Sankeerna placement for example
(b) PDP placement for example

Figure 6 *Sankeerna* and Publice Domain Placer (PDP) Placements of example

Once the placer completes the placement of tree pointed by Ua356 as root, it starts placing the tree pointed by cell Ua358. Now the cells marked 22 to 26 placed. This completes the placement of complete circuit. Primary Inputs and Primary Outputs are re-adjusted after placing all the cells. The final placement is shown in Figure 6(a). The cell name, sequence numbers and pin locations of the cell are shown in the Figure 6(a). These diagrams are taken from Cadence® SOC Encounter® back end tool [58]. The placement given by the Public Domain Placer (PDP) [59, 60, 41, 61] for this example is also shown in Figure 6(b). The layouts of these placements after carrying out detailed routing with Cadence® SOC Encounter® [58] are shown in Figure 7. The results are shown in table 1, Serial Number 24, as "example".

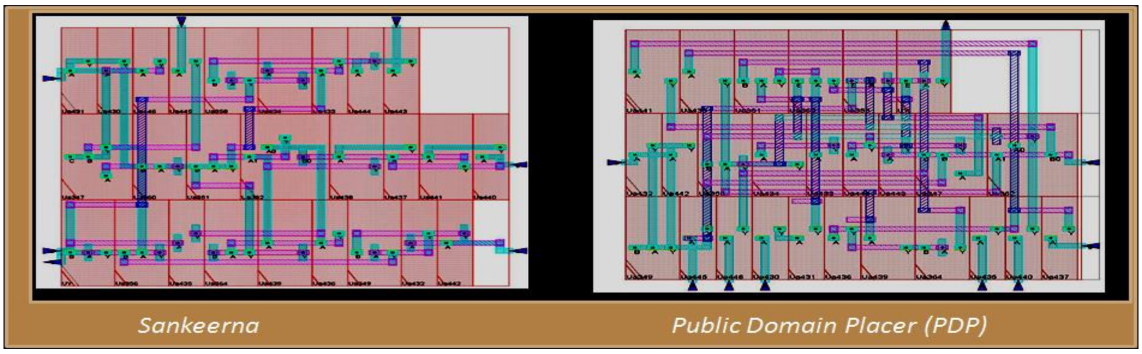

Figure 7 *Sankeerna* & PDP layout of example

The experimental set up to evaluate *Sankeerna* using benchmark circuits is explained in the next section.

VI. TEST SETUP

In this section, we describe the test set up used to evaluate *Sankeerna*. The test set up is shown in Figure 8.

The benchmark circuits are taken in the form of a PLA. Some of them are from MCNC benchmarks. We also added few other circuits like multiplexers, adders and multipliers. We have used 0.13 micron, 8 metal layers, standard cell technology library for these experiments. Public domain SIS synthesizer [52] is used for synthesizing the benchmark circuits.

We created the library file required for SIS in genlib format using the information from the data sheets and ".lef" files of this particular technology. Three files namely (a) delay file (b) Cell mapping file (c) BLIF file are generated from SIS. The delay file consists of node name and slack at each node. Only block delays and fan out delays are considered. No wire delay or wire load models are used. This delay file is created using the information generated by "print_delay" command of SIS. The cell mapping file consists of node name and mapped library cell name. This file is created using the information generated by "print_gate" command of SIS. The BLIF file is created using "write_blif" command of SIS. The BLIF output is then converted into Bookshelf format using the public domain tools available at the web site [59, 60, 41, 61] using the utility "blif2book-Linux.exe filename.blif filename". Using 0.13 micron standard cell technology files, *Sankeerna* generates a file in bookshelf format using the cell dimensions and pin locations of the given technology library. This file is used for placement by *Sankeerna* and also by Public Domain Placer (PDP) [59].

Bench marks are placed in case of PDP flow using "time LayoutGen-Lnx32.exe -f filename.aux –AR 1.5 -saveas filename" [59]. The width to height ratio is 3:2 which is same for *Sankeerna*. *Sankeerna* gives the placement output in ".def" format [62] (".def file"). The mapped netlist is given out in the form of structural verilog file (".v file"). Cadence® SOC Encounter® v06.10-p005_1 [58] is used for routing and calculating the delay of the placement produced. The verilog (.v file) and placement (.def file) are read into SOC Encounter®. The 0.13 micron standard cell technology files, consisting of ".lef" files and timing library files ".tlf" are read into SOC encounter. We did a trial route and then detailed routing using Cadence NanoRoute® v06.10-p006 [58].

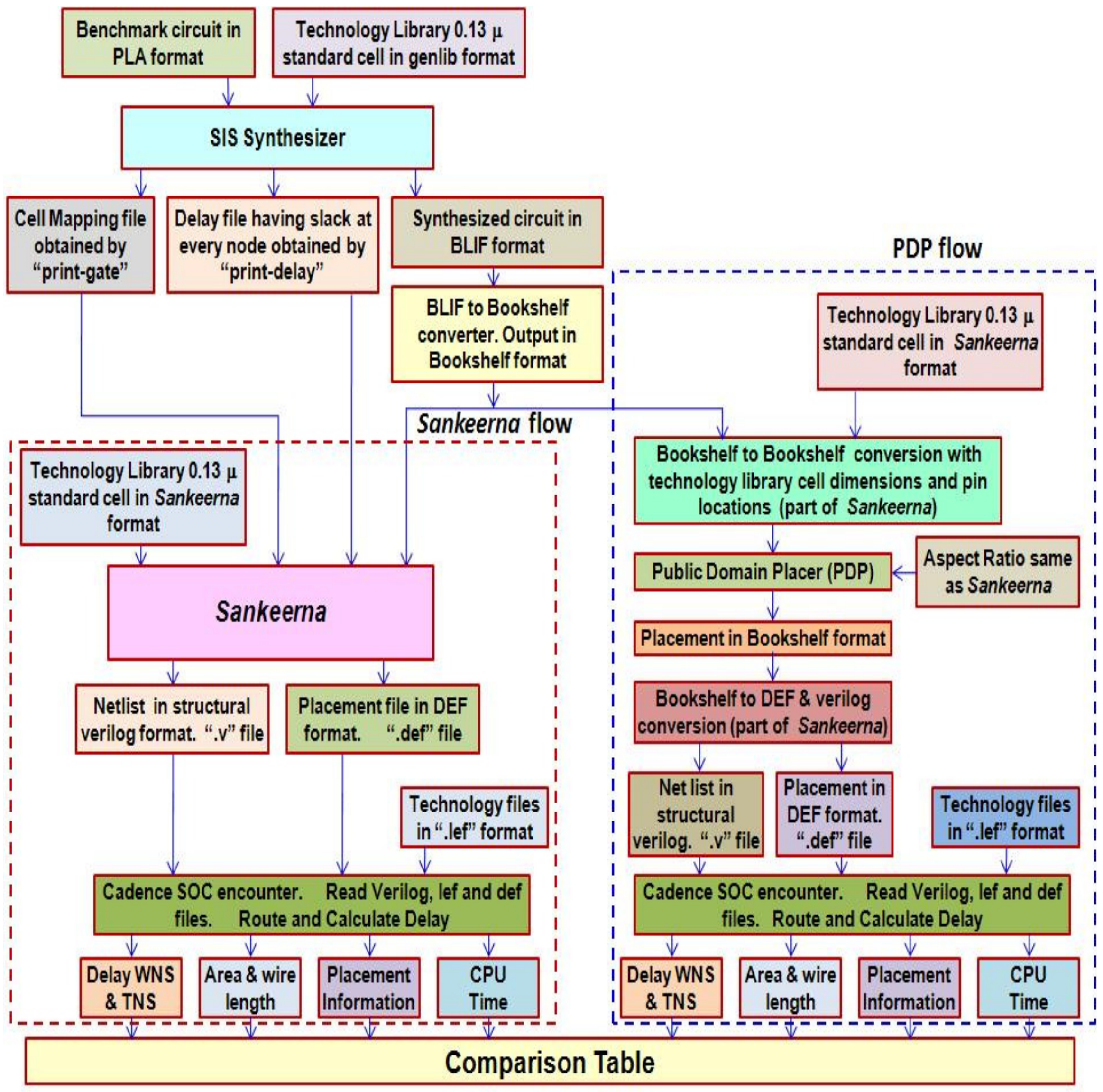
Benchmark circuit in PLA format
Technology Library 0.13 µ standard cell in genlib format
SIS Synthesizer
Cell Mapping file obtained by "print-gate"
Delay file having slack at every node obtained by "print-delay"
Synthesized circuit in BLIF format
BLIF to Bookshelf converter. Output in Bookshelf format
PDP flow
Sankeerna flow
Technology Library 0.13 µ standard cell in Sankeerna format
Sankeerna
Netlist in structural verilog format. ".v" file
Placement file in DEF format. ".def" file
Technology files in ".lef" format
Cadence SOC encounter. Read Verilog, lef and def files. Route and Calculate Delay
Delay WNS & TNS
Area & wire length
Placement Information
CPU Time
Technology Library 0.13 µ standard cell in Sankeerna format
Bookshelf to Bookshelf conversion with technology library cell dimensions and pin locations (part of Sankeerna)
Public Domain Placer (PDP)
Aspect Ratio same as Sankeerna
Placement in Bookshelf format
Bookshelf to DEF & verilog conversion (part of Sankeerna)
Net list in structural verilog. ".v" file
Placement in DEF format. ".def" file
Technology files in ".lef" format
Cadence SOC encounter. Read Verilog, lef and def files. Route and Calculate Delay
Delay WNS & TNS
Area & wire length
Placement Information
CPU Time
Comparison Table

Figure 8 Test setup showing *Sankeerna* VLSI Design Flow

The delays Worst Negative Slack (WNS) and Total Negative Slack (TNS) "from all inputs to all outputs" are noted. The CPU time for detailed routing is also noted. Various other characteristics of placed circuits namely standard cell area, core area, wire length and Worst Negative Slack (WNS) are noted. Results for various benchmarks are shown in Table 1. We compared the results produced by *Sankeerna* with a Public Domain Placer (PDP) [59]. The PDP flow is also shown in Figure 8. The PDP uses the benchmark circuits in bookshelf format. The BLIF file generated by SIS is converted into bookshelf format with cell dimensions and pin locations as per the 0.13 micron standard cell library. We have used same aspect ratio for *Sankeerna* and PDP. The aspect ratio for these experiments was 0.4 for height and 0.6 for width. To have a fair comparison, the Primary Inputs and Primary Outputs are placed at the boundary of the core for both placements produced by *Sankeerna* and PDP. The output from PDP [59] is in bookshelf format (.pl, .scl files). This is converted into ".def" format. The netlist is generated in the form of structural verilog file. All the utilities used to convert the above file formats are developed as software package of *Sankeerna*. The verilog, ".def", the technology file (.lef file) and timing files (.tlf files) are read into Cadence® SOC Encounter® [58]. The detailed routing and delay calculations are carried out. A Linux machine with dual Intel® Pentium® 4 CPU @3.00GHz and 2 GB memory was used for running *Sankeerna* and PDP. For running SOC Encounter® [58], Sun Microsystems SPARC Enterprise® M8000 Server with 960 MHz CPU and 49 GB memory was used. The results are tabulated in Table 1.

VII. RESULTS AND DISCUSSION

The Table 1 shows the results of the placed circuits using existing Public Domain Placer (PDP) [59] and *Sankeerna*.

Table 1 Test Results

Sl No	Name	Std Cell Area	Pre-place WNS	*Sankeerna*				PDP				% of area increase over Std Cell		% improvements over PDP		
				Area	WNS	Wire Len-gth	DRC violations	Area	WNS	Wire Length	DRC violations	S	PDP	Ar ea	Wire Leng th	WN S
1	5xp1	912	-1.06	927	-1.274	2102	0	1013	-1.752	4037	0	2	11	8	92	38
2	9sym	1482	-1.19	1512	-1.568	5256	0	1647	-1.858	8125	0	2	11	8	55	18
3	alu4	6723	-2.41	6773	-3.237	33296	0	7470	-5.528	89520	13434	1	11	9	169	71
4	b12	536	-0.71	550	-0.850	1195	0	596	-0.891	1476	0	3	11	8	24	5
5	clip	1268	-1.51	1290	-1.762	3371	0	1409	-2.099	6170	0	2	11	8	83	19
6	cm82a	156	-0.54	173	-0.603	204	0	173	-0.598	233	0	11	11	0	14	-1
7	comp	1190	-1.24	1222	-1.467	2943	0	1322	-1.904	5391	0	3	11	8	83	30
8	con1	139	-0.42	148	-0.489	198	0	155	-0.513	249	0	6	11	5	26	5
9	cordic	732	-0.98	754	-1.152	1406	0	813	-1.250	2645	0	3	11	7	88	9
10	count	1275	-0.97	1290	-1.238	3535	0	1416	-1.466	5863	0	1	11	9	66	18
11	e64	7002	-0.96	7031	-1.657	44024	0	7780	-3.316	94434	17461	0	11	10	115	100
12	ex5	3057	-0.97	3133	-1.334	13181	0	3397	-1.899	23916	0	2	11	8	81	42
13	misex1	497	-0.76	509	-0.902	999	0	553	-0.967	1536	0	2	11	8	54	7
14	misex2	884	-0.64	927	-0.765	2284	0	983	-0.941	3475	0	5	11	6	52	23
15	mux8-1	175	-0.59	188	-0.655	224	0	194	-0.716	310	0	8	11	3	38	9
16	o64	1514	-0.68	1558	-0.816	2603	0	1682	-1.026	7562	0	3	11	7	191	26
17	rd53	341	-0.71	356	-0.827	640	0	379	-0.854	739	0	4	11	6	15	3
18	rd73	1044	-1.22	1059	-1.441	2715	0	1160	-1.834	4622	0	1	11	9	70	27
19	rd84	949	-1.06	974	-1.347	2513	0	1054	-1.702	4351	0	3	11	8	73	26
20	sao2	1008	-1.26	1059	-1.390	2564	0	1120	-1.792	4359	0	5	11	5	70	29
21	squar5	475	-0.85	492	-0.962	943	0	528	-0.995	1314	0	4	11	7	39	3
22	t481	217	-0.65	231	-0.699	228	0	241	-0.762	285	0	6	11	4	25	9
23	Z9sym	745	-1.23	784	-1.495	1976	0	828	-1.836	2952	0	5	11	5	49	23
24	example	115	-0.60	127	-0.651	137	0	168	-0.605	241	0	10	45	24	76	-7
25	add556	1516	-1.23	1558	-1.560	4737	0	1684	-1.984	7529	0	3	11	7	59	27
26	mul5510	8553	-3.02	8626	-4.765	44913	0	9504	-7.325	114480	17586	1	11	9	155	54
27	mul448	2161	-1.89	2203	-2.520	7008	0	2401	-3.039	14457	0	2	11	8	106	21
28	add667	2348	-1.55	2390	-2.140	8079	0	2608	-2.741	14472	0	2	11	8	79	28
29	add889	5708	-1.87	5742	-3.068	26907	0	6342	-4.172	62149	2398	1	11	9	131	36
30	add778	3422	-1.87	3486	-2.786	12462	0	3794	-3.294	22303	0	2	11	8	79	18
										Average		3	12	8	75	24

There is an average improvement of 24% in delay (Worst Negative Slack WNS), 8% in area and 75% in wire length after detailed routing with Nano-route of Cadence [58] when compared to Public Domain Placer (PDP). *Sankeerna* uses only 3% of area extra over the standard cell area as shown under "S" in Table 1 where as PDP uses 12%. In case of bigger bench marks, namely, alu4, e64, mul5510 and add889, there are thousands of DRC violations in case of PDP. This is due to increased usage of wire length. So those placements are not useful.

To compare *Sankeerna* with a commercial tool, we conducted the following experiment. We placed the benchmark circuit "alu4" using *Sankeerna* and did the detailed routing and delay calculation as mentioned earlier. Then the results given by *Sankeerna* are noted. We specified the same dimensions of width and height in the Commercial Tool Placer (CTP) after reading the verilog file into the tool. We then ran timing driven placement of commercial tool. We then carried out detailed routing. CPU time used for *Sankeerna* and Commercial Tool Placer (CTP) are given in Table 2. SOC Encounter® took 2:27 CPU time for detailed routing using Nanoroute® [58] for *Sankeerna* placement without any DRC violations. SOC Encounter® took 6:45 CPU time for detailed routing using Nanoroute for CTP's timing driven placement with 144 DRC violations in Metal layer 1. The layouts produced for

Sankeerna placement and CTP's placement are shown in Figure 9. The black marks in CTP layouts in the Figure 9 are the DRC violations. These are shown separately in the same Figure (extreme right block). Since there are so many DRC violations, the layout is not useful.

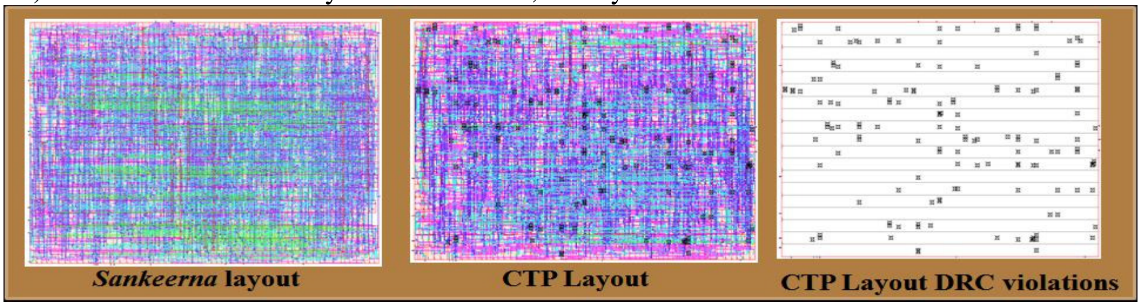

Figure 9 Layouts of ALU4 with *Sankeerna* & CTP

Table 2 shows results of other benchmarks comparing with Commercial Tool timing driven Placer (CTP).

Table 2 Comparing *Sankeerna* with Commercial Tool Placer (CTP)

Sl No	Name	Std Cell Area	Core Area	Floor Plan Dimension		*Sankeerna*				CTP			
				Width	Height	CPU Time	WNS	TNS	DRC viola-tions	CPU Time	WNS	TNS	DRC viola-tions
1	alu4	6723	6773	96.60	70.11	2:27	-3.24	-23.22	0	6:45	-3.35	-21.58	144
2	e64	7002	7031	100.28	70.11	8:21	-1.66	-93.93	0	12:57	-1.33	-73.01	127
3	mul5510	8553	8626	111.32	77.49	1:17	-4.77	-39.70	0	10:04	-4.47	-35.11	283
4	add889	5708	5742	91.54	62.73	2:42	-3.07	-21.97	0	10:08	-2.67	-19.29	103

For the same floor plan dimensions, CTP took more CPU time and could not produce final usable placement due to several DRC violations as shown in the last column of Table 2. *Sankeerna* placements are always routed without DRC violations, because wire planning is already done using shortest wires. This avoids iterations between placement and detailed routing steps. Even though, WNS and TNS values shown in the Table are comparable, because of DRC violations, the layout produced by CTP is not useful; hence delay values have no meaning in case of CTP. So our method took much less time and produced better results when compared to Commercial Tool Placer (CTP).
Pre-place delay is computed without taking wire delays into considerations. We have not used wire load models in SIS, because wire load models are not good estimates of final delay [2]. The post route delay includes delay due to wires and is the final delay of the circuit. We have compared pre-place delay with post route delay in Table 3.

Table 3 Pre-place delay versus post routed delay comparison

Sl No	Name	Preplace	*Sankeerna*	PDP	*Sankeerna*	PDP
		WNS	WNS	WNS	WNS Diff %	WNS Diff %
1	alu4	-2.41	-3.36	-5.50	39.39	128.50
2	e64	-0.96	-1.72	-3.30	78.61	242.99
3	ex5	-0.97	-1.32	-1.85	36.37	91.19
4	mul5510	-3.02	-4.77	-6.84	57.57	126.12
5	mul448	-1.89	-2.52	-3.06	33.69	62.33
6	add889	-1.87	-3.07	-4.13	64.24	120.82
7	add778	-1.87	-2.79	-3.41	49.22	82.65
					51.30	**122.09**

The percentage difference of Worst Negative Slack of *Sankeerna* placements is much less (51.30% versus 122.09%) when compared to Public Domain Placer's (PDP's) values. This is due to the fact that PDP uses more wire when compared to *Sankeerna*.
Cadence® SOC Encounter® [58] has the facility called "Trial route", which performs quick global and detailed routing creating actual wires, estimating routing related congestion and capacitance values. We did trial route of the benchmarks and noted wire lengths, WNS and TNS values for *Sankeerna* and PDP placements. The results are compared in Table 4.

Table 4 Trial route Versus Detail (Nano) route [58], wire length and delay comparison

Sl No	Name	Trial Route				Nano Route				*Sankeerna* Trial Vs Nano		PDP Trial vs Nano	
		Sankeerna		PDP		*Sankeerna*		PDP					
		Wire-length	WNS	Wire length	WNS	Wire length	WNS	Wire length	WNS	Wire-length Diff %	WNS Diff %	Wire Length Diff %	WNS Diff %
1	alu4	33418	-3.26	73683	-4.90	33296	-3.24	89520	-5.53	-0.36	-0.61	21.49	12.82
2	e64	43530	-1.69	79155	-2.80	44024	-1.66	94434	-3.32	1.14	-2.18	19.30	18.43
3	ex5	13486	-1.30	23531	-1.90	13181	-1.33	23916	-1.90	-2.27	2.38	1.64	-0.05
4	mul5510	44440	-4.63	97929	-6.30	44913	-4.77	114480	-7.33	1.06	3.03	16.90	16.27
5	mul448	7147	-2.44	14442	-3.03	7008	-2.52	14457	-3.04	-1.94	3.11	0.10	0.20
6	add889	27108	-3.01	54723	-4.03	26907	-3.07	62149	-4.17	-0.74	1.83	13.57	3.65
7	add778	12798	-2.77	22371	-3.26	12462	-2.79	22303	-3.3	-2.62	0.76	-0.30	1.10
									Average %	-0.82	1.19	10.39	7.49

The percentage differences from trial route to detailed (Nano) route [58] are shown in the last 4 columns of Table 4 for both *Sankeerna* and PDP. There is decrease in wire length by 0.82%, 1.19% increase in WNS in case of *Sankeerna*. The PDP placements took 10.39% more wire, WNS increased by 7.49%. So the Trial Route produced delays are good estimates in case of *Sankeerna* when compared to PDP placements. So we can get quick estimate of delay for *Sankeerna* produced placements. All these advantages add towards tighter coupling of VLSI design flow as envisaged in [4] to evolve Synergistic Design Flow. *Sankeerna* scales well as the complexity increases, because we are using constructive placement. The placement was found to converge after routing in all the test cases we have tried, that is, layout generated was without DRC violations after routing. To prove this point, we have taken bigger benchmarks out of Table 1 and improvements obtained are shown in Table 5.

Table 5 As complexity increases, *Sankeerna* performs better over PDP.

Sl No	Name	Std Cell Area	Pre-place WNS	*Sankeerna*				PDP				% Area Increase over Std Cell area		% Improvements		
				Area	WNS	Wire Len-gth	DRC violati ons	Area	WNS	Wire Length	DRC violatio ns	S	PDP	Ar ea	Wire Length	WNS
1	alu4	6723	-2.41	6773	-3.237	33296	0	7470	-5.528	89520	13434	0.7	11.1	9.3	168.9	70.8
2	e64	7002	-0.96	7031	-1.657	44024	0	7780	-3.316	94434	17461	0.0	11.0	9.6	114.5	100.1
3	ex5	3057	-0.97	3133	-1.334	13181	0	3397	-1.899	23916	0	2.0	11.0	7.8	81.4	42.4
4	mul551	8553	-3.02	8626	-4.765	44913	0	9504	-7.325	114480	17586	1.0	11.0	9.2	154.9	53.7
5	mul448	2161	-1.89	2203	-2.520	7008	0	2401	-3.039	14457	0	2.0	11.0	8.2	106.3	20.6
6	add667	2348	-1.55	2390	-2.140	8079	0	2608	-2.741	14472	0	2.0	11.0	8.4	79.1	28.1
7	add889	5708	-1.87	5742	-3.068	26907	0	6342	-4.172	62149	2398	1.0	11.0	9.5	131.0	36.0
8	add778	3422	-1.87	3486	-2.786	12462	0	3794	-3.294	22303	0	2.0	11.0	8.1	79.0	18.2
											Average	1.3	11.0	8.8	114.4	46.2

As can be seen from the Table 5, *Sankeerna* took only 1.3% extra area over the standard cell area where as PDP took 11% extra area. The 1.3% extra area is left at the right edges of all rows. This is the bare minimum area required to make the placement possible. The area improvement of *Sankeerna* over PDP is 8.8%. The most interesting fact is that wire length is 114.4% more for PDP when compared to *Sankeerna*. WNS improved by 46.2%. There are several DRC violations after detailed routing in case of PDP. Hence those layouts are not useful. *Sankeerna* used bare minimum area and produced better timings which were routable. Area, wire length and delay are interrelated. If we use minimum area, as is the case with *Sankeerna*, we require less length wire. This in turn leads to minimum delay. So using *Sankeerna* flow avoids design iterations, because the flow is tightly coupled from synthesis to layout. Thus *Sankeerna* produced compact layouts which are better in delay when compared to Public Domain Placer (PDP) and Commercial Tool's timing driven Placer (CTP).

VIII. CONCLUSIONS AND FUTURE SCOPE

We now summarise the features of *Sankeerna* for fitting it into a Synergistic Design Flow (SDF) [4] as mentioned in Section 3.

The first requirement was (a) linear time complexity with respect to number of cells in the circuit. The *Sankeerna* placement algorithms have linear time complexity after construction of the tree. In a tightly coupled Synergistic Design Flow (SDF) [4], trees are already built by synthesizer which can be directly used by the placer. So there is no need to construct them separately. Due to linear time complexity, *Sankeerna* scales well as the circuits become bigger.

The second requirement was (b) awareness of synthesis and routing assumptions and expectations, that is, tight coupling of synthesis and routing as mentioned in [4]. We have used logic dependency and timing information from synthesizer. Using this information, it properly guides the placement as mentioned in [46, 28, 29, 30, 47]. As shown in Table 3, pre-place to post routed delay variation for *Sankeerna* was 51.30% when compared to 122.09% for Public Domain Placer (PDP) [59]. The values vary from 39.39% to 78.6% for *Sankeerna*. Where as the variation for PDP was, from 62.33% to 242.99% based on the circuit. So *Sankeerna* is more coupled to synthesizer's estimates when compared to PDP. *Sankeerna* placements were always routed without DRC violations as shown in Table 1 and 2. Where as PDP has thousands of violations for bigger circuits even after using 11% extra space when compared to *Sankeerna* which used only 0% to 1% extra over the area calculated by the synthesizer, which is bare minimum over the standard cell area. For the same floor plan dimensions, Commercial Tool's timing driven Placer (CTP) produced hundreds of DRC violations as shown in Table 2 when compared to zero DRC violations for *Sankeerna*. In *Sankeerna* routability is achieved without white space allocation, because placements produced by *Sankeerna* use minimum length wires. As mentioned in Section 2, white space allocation increases area which in turn increase wire length. In conclusion, the placements produced by *Sankeerna* were always routable because it uses minimum wire when compared to PDP and CTP.

The third requirement was (c) achieving minimum area and delay. Area increase was only 1.3% over the standard cell area which is calculated by the synthesizer. This value for PDP was 11%. As shown in Table 5, *Sankeerna* performed better as the complexity increased when compared to Public Domain Placer (PDP) [59]. Wire length improved by 114.4% and delay by 46.2% when compared to PDP.

The fourth requirement was (d) placement produced should be routable without Design Rule Check (DRC) violations, that is, wire planning has to be done during placement. As shown in Table 5, PDP could not produce usable placements due congestion and resulted in thousands of DRC violations in four cases out of 8 test cases. So design flow did not converge in case of PDP. It is unpredictable, because, in some cases it converged. In case of *Sankeerna*, it always converged and convergence is predictable due to minimum wire length and proper wire planning done during placement. As shown in Table 2, same non-convergence and unpredictability was noticed in case of Commercial Tool's Placer (CTP).

The fifth requirement was (e) delay of final layout should be predictable with trial routes. As shown in Table 4, for *Sankeerna*, wire length decreased by 0.82% from trial route to detailed route, where as it increased by 10.39 % for PDP. The delay increase was 1.19% for *Sankeerna*, where as it is 7.49% for PDP. Thus, the wire planning done by *Sankeerna* was maintained after routing, whereas it varied in the case of PDP. Trial route [58] was good estimate for *Sankeerna*. So there is convergence and tight coupling between placement and routing in case of *Sankeerna*.

The sixth requirement was (f) placer should smoothly interface with synthesis and routing tools. As shown in Figure 8, the test set up showing *Sankeerna* design flow, *Sankeerna* smoothly interfaces with the existing synthesizer and router. We have not used any wire load models during synthesis as it was demonstrated that they were not useful [2]. The slack at each node and cell mapping information were already available with the synthesizer. The router was interfaced through verilog netlist and "def" [62] file for placement information. Pre-place and trial route delay calculations were done by the commercial tool [58], which were good estimators in case of *Sankeerna* for a real standard cell technology library. As can be seen from the experiments, there were no design iterations among synthesis, placement and routing in case of *Sankeerna* to achieve the results shown.

Area, wire length and delay calculations of Pre-place, trial and post route were done by the Commercial tool [58]. This validates that there is no error in measuring these values while conducting these experiments.

The features and effectiveness comparison of *Sankeerna* with other published placement techniques is elaborated here. In *Sankeerna* the cells which are logically dependent are placed closer to each other [29], whereas in other placement algorithms the cells are randomly scattered and create zigzags and criss-crosses that leads to increase in congestion, wire length and delays. The random scattering of the cells even leads to the unpredictability in the final layout that result into non-convergent iterations. Because wires are shorter in our placement and wires are planned by selecting closest locations during placement, congestion is less and detailed routing always gets completed using minimum area for wires. This automatically leads to minimum delays. The most critical paths automatically get higher priority, without going in for path based placement, which grows exponentially with circuit complexity and is computationally expensive. As it can be seen from Figure 5, *Sankeerna* first establishes the most critical path and rest of the logic is placed around it based on the logic dependency. This is similar to the denser path placement of [30]. So the most critical path is placed, along physically and delay wise shortest path as mentioned in Section 3. Since our method is constructive, it scales well for bigger circuits. We are planning to test with bigger test cases in future. The Circuit is naturally partitioned when trees are built, rooted by Primary Outputs (POs) by *Sankeerna*. So there is no additional burden of extracting cones as in [27, 29] or partitioning the circuit as is the case in most of the placers. Global signal flow is kept in mind all through the placement by using Depth First Search (DFS) for all sub trees rooted at various levels of logic, unlike other placement methods, which randomly scatter the cells. Trial route [58] can be used for quick estimate of delay which will be good estimate in case of *Sankeerna* as explained earlier. As mentioned in [2], using wire load models misleads whole design process resulting in non-convergence. So *Sankeerna* flow does not use wire load models. *Sankeerna* flow is always convergent and tightly coupled, which gives estimates of area, wire length and delay using existing layout tools like Trial route of Cadence's SOC Encounter® [58], which are not far away from the values obtained after detailed routing. Thus *Sankeerna* approach is useful towards evolving Synergistic Design Flow (SDF), which is to create iteration loops that are tightly coupled at the various levels of design flow as mentioned in [4].

ACKNOWLEDGEMENTS

We thank Dr. K.D. Nayak who permitted and guided this work to be carried out in ANURAG. We also thank members of ANURAG who reviewed the manuscript. Thanks are due to Mrs. D. Manikyamma and Mr. D. Madhusudhan Reddy for preparation of the manuscript.

REFERENCES

[1] Randal E. Byrant, et al., (2001), "*Limitations and Challenges of Computer-Aided Design Technology for CMOS VLSI*", *Proceedings of the IEEE*, Vol. 89, No. 3, pp 341-65.

[2] Gosti, W., et al., (2001), "*Addressing the Timing Closure Problem by Integrating Logic Optimization and Placement*", *ICCAD 2001 Proceedings of the 2001 IEEE/ACM International Conference on Computer-aided design*, San Jose, California , pp 224-231.

[3] Shahookar K & Mazumder P, (1991), "*VLSI cell placement techniques*" *ACM Computing Surveys*, Vol. 23, No. 2.

[4] Kurt Keutzer., et al., (1997), *"The future of logic synthesis and physical design in deep-submicron process geometries*", *ISPD '97 Proceedings of the international symposium on Physical design*, ACM New York, NY, USA, pp 218-224.

[5] Coudert, O, (2002), "*Timing and design closure in physical design flows*", *Proceedings. International Symposium on Quality Electronic Design (ISQED '02)*, pp 511 – 516.

[6] Wilsin Gosti , et al., (1998), "*Wireplanning in logic synthesis*", *Proceedings of the IEEE/ACM international conference on Computer-aided design*, San Jose, California, USA, pp 26-33.

[7] Yifang Liu, et al., (2011), "*Simultaneous Technology Mapping and Placement for Delay Minimization", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 30 No. 3, pp 416–426.

[8] Pedram, M. & Bhat, N, (1991), "*Layout driven technology mapping*", *28th ACM/IEEE Design Automation Conference,* pp 99 – 105.

[9] Salek, A.H., et al., (1999), "*An Integrated Logical and Physical Design Flow for Deep Submicron Circuits", IEEE Transactions on* Computer-Aided Design of Integrated Circuits and Systems, Vol. 18,No. 9, pp 1305–1315.

[10] Naveed A. Sherwani, (1995), "*Algorithms for VLSI Physical Design Automation*", Kluwer Academic Publishers, Norwell, MA, USA.

[11] Sarrafzadeh, M., & Wong, C.K., (1996), "*An introduction to VLSI Physical Design*", The McGraw-Hill Companies, New York.

[12] Jason Cong, et al., (2005), "*Large scale Circuit Placement*", *ACM Transactions on Design Automation of Electronic Systems*, Vol. 10, No. 2, pp 389-430.

[13] Yih-Chih Chou & Young-Long Lin, (2001), "*Performance-Driven Placement of Multi-Million-Gate Circuits*", *ASICON 2001 Proceedings of 4th International Conference on ASIC*, Shanghai, China, pp 1-11.

[14] Yao-Wen Chang, Zhe-Wei Jiang and Tung-Chieh Chen, (2009), "*Essential Issues in Analytical Placement Algorithms", Information and Media Technologies*, Vol. 4, No. 4, pp.815-836

[15] Bunglowala, A. & Singhi, B.M., (2008), "*Performance Evaluation and Comparison and Improvement of Standard Cell Placement Techniques in VLSI Design*", *First international conference on Emerging Trends in Engineering and Technology,* Nagpur, Maharashtra, 468 – 473.

[16] B. Sekhara Babu, et al (2011), "*Comparison of Hierarchical Mixed-Size Placement Algorithms for VLSI Physical Synthesis*", *CSNT '11 Proceedings of the 2011 International Conference on Communication Systems and Network Technologies*, IEEE Computer Society Washington, DC, USA, pp 430-435.

[17] Mehmet Can Yildiz & Patrick H. Madden, (2001), "*Global objectives for standard cell placement*", *GLSVLSI '01 Proceedings of the 11th Great Lakes symposium on VLSI*, ACM New York, NY, USA, pp 68 – 72.

[18] C. J. Alpert, et al., (1997), "*Quadratic Placement Revisited*", *34th ACM/IEEE Design Automation Conference*, Anaheim, pp 752-757.

[19] N. Viswanathan et al., (2007), "*FastPlace 3.0: A Fast Multilevel Quadratic Placement Algorithm with Placement Congestion Control*", *ASP-DAC '07 Proceedings of the 2007 Asia and South Pacific Design Automation Conference*, IEEE Computer Society Washington, DC, USA, pp 135-140.

[20] Spindler, P, et al., (2008), "*Kraftwerk2—A Fast Force-Directed Quadratic Placement Approach Using an Accurate Net Model*", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* Vol. 27 no. 8, 1398 – 1411.

[21] Rexford D. Newbould & Jo Dale Carothers , (2003), "*Cluster growth revisited: fast, mixed-signal placement of blocks and gates*", *Southwest Symposium on Mixed Signal Design*, pp 243 – 248.

[22] Carl Sechen & Alberto Sangiovanni-Vincentelli, (1985), "*The TimberWolf Placement and Routing Package*", *IEEE Journal of Solid-State Circuits*, vol. SC-20, No. 2, pp 510-522.

[23] Khorgade, M. et al., (2009), "*Optimization of Cost Function with Cell Library Placement of VLSI Circuits Using Simulated Annealing*", *2nd International Conference on Emerging Trends in Engineering and Technology (ICETET)*, Nagpur, India, pp 173 – 178.

[24] Yoshikawa, M. & Terai, H., (2005), "*A GA-based timing-driven placement technique*", *Sixth international conference on Computational Intelligence and Multimedia Applications*, pp 74 – 79.

[25] Andrew Kennings & Kristofer P. Vorwerk, (2006), "*Force-Directed Methods for Generic Placement*", *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, Vol. 25, N0. 10, pp 2076-2087.

[26] Chen Li et al., (2007), "*Routability-Driven Placement and White Space Allocation*", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Volume: 26 Issue:5, pp 858 – 871.

[27] Yu-Wen Tsay, et al., (1993), *"A Cell Placement Procedure that Utilizes Circuit Structural Properties*", *Proceedings of the European Conference on Design Automation*, pp 189-193.

[28] Chanseok Hwang & Massoud Pedram, (2006), "*Timing-Driven Placement Based on Monotone Cell Ordering Constraints*", *Proceedings of the 2006 Conference on Asia South Pacific Design Automation: ASP-DAC 2006*, Yokohama, Japan, pp 201-206.

[29] Cong, J. & Xu, D, (1995), " *Exploiting signal flow and logic dependency in standard cell placement*", *Proceedings of the Asian and South Pacific Design Automation Conference,* pp 399 – 404.

[30] Ioannis Fudos et al,(2008), "*Placement and Routing in Computer Aided Design of Standard Cell Arrays by Exploiting the Structure of the Interconnection Graph", Computer-Aided Design & Applications*, CAD Solutions, LLC, Canada, http://www.cadanda.com/, Vol. 5(1-4), pp 325-337.

[31] Andrew B. Kahng & Qinke Wang, (2004), "*An analytic placer for mixed-size placement and timing-driven placement*", *Proceedings of International Conference on Computer Aided Design*, pp 565-572.

[32] Jun Cheng Chi, et al., (2003), "*A New Timing Driven Standard Cell Placement Algorithm*", *Proceedings of International Symposium on VLSI Technology, Systems and Applications,* pp 184-187.

[33] Swartz, W., & Sechen, C., (1995), "*Timing Driven Placement for Large Standard Cell Circuits*", *Proc. ACM/IEEE Design Automation Conference*, pp 211-215.

[34] Tao Luo, et al., (2006), "*A New LP Based Incremental Timing Driven Placement for High Performance Designs*", *DAC '06 Proceedings of the 43rd Annual Design Automation Conference, ACM New York*, NY, USA, pp 1115-1120.

[35] Wern-Jieh Sun & Carl Sechen, (1995), "*Efficient and effective placement for very large circuits*",. *IEEE Transactions on CAD of Integrated Circuits and Systems, Vol. 14* No. 3, pp 349-359.

[36] Marek-Sadowska, M. & Lin, S.P. (1989), "Timing driven placement" IEEE international conference on Computer-Aided Design, Santa Clara, CA , USA , pp 94 - 97

[37] Wilm E. Donath, et al., (1990), "*Timing driven placement using complete path delays*", *Proceedings of the 27th ACM/IEEE Design Automation Conference*, ACM New York, NY, USA, pp 84 – 89.

[38] Bill Halpin, et al, (2001), "*Timing driven placement using physical net constraints*", *Proceedings of the 38th annual Design Automation Conference*, ACM New York, NY, USA, pp 780 – 783.

[39] Xiaojian Yang, et al., (2002), "*Timing-driven placement using design hierarchy guided constraint generation*" *Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design*, ACM New York, NY, USA, pp 177-180.

[40] Riess, B.M. & Ettelt, G.G, (1995), "*SPEED: fast and efficient timing driven placement*", *IEEE International Symposium on Circuits and Systems,* Seattle, WA , USA, vol.1, pp 377 – 380.

[41] Saurabh Adya and Igor Markov, (2003), "*On Whitespace and Stability in Mixed-size Placement and Physical Synthesis*", *International Conference on Computer Aided Design (ICCAD)*, San Jose, pp 311-318.

[42] Taraneh Taghavi, et al, (2006), "*Dragon2006: Blockage-Aware Congestion-Controlling Mixed-Size Placer*", *ISPD '06 Proceedings of the 2006 international symposium on Physical design*, ACM New York, NY, USA, pp 209 – 211.

[43] Yi-Lin Chuang, et al., (2010), "*Design-hierarchy aware mixed-size placement for routability optimization*", *IEEE/ACM International Conference on Computer-Aided Design (ICCAD),* San Jose, CA, USA, pp 663 – 668.

[44] Xiaojian Yang et al., (2002), "*A standard-cell placement tool for designs with high row utilization*", *Proceedings of IEEE International Conference on Computer Design: VLSI in Computers and Processors*, pp 45 – 47.

[45] C. Chang, J. Cong, et al., (2004), "*Optimality and Scalability Study of Existing Placement Algorithms*", *IEEE Transactions on Computer-Aided Design*, Vol.23, No.4, pp.537 – 549.

[46] Santeppa Kambham & Krishna Prasad K.S.R, (2011), "*ANUPLACE: A Synthesis Aware VLSI Placer to minimize timing closure*", *International Journal of Advances in Engineering & Technology*, Vol. 1, Issue 5, pp. 96-108.

[47] Santeppa Kambham et al., (2008), "*New VLSI Placement algorithms to minimize timing closure problem*", *International conference on Emerging Microelectronics and Interconnection Technology (EMIT-08),* IMAPS, Bangalore, India.

[48] Brayton R K, et al., (1990), "*Multilevel Logic Synthesis*", *Proceedings of the IEEE*, Vol. 78, No. 2, pp-264-300.

[49] Brayton R K, et al.,(1987), "*MIS: A Multiple-Level Logic Optimization System*", *IEEE Transactions on Computer Aided Design*, Vol.6, No.6, pp-1062-1081.

[50] Rajeev Murgai, et al.,(1995), "*Decomposition of logic functions for minimum transition activity*", *EDTC '95 Proceedings of the European conference on Design and Test*, pp 404-410.

[51] Fujita, M. & Murgai, R, (1997), "*Delay estimation and optimization of logic circuits: a survey*", *Proceedings of Asia and South Pacific Design Automation Conference,* Chiba,Japan, pp 25 – 30.
[52] Sentovich, E.M., et al., (1992), "*SIS: A System for Sequential Circuit Synthesis*", Memorandum No. UCB/ERL M92/41, Electronics Research Laboratory, University of California, Berkeley, CA 94720.
[53] M. Fischer & M. Paterson, (1980), "*Optimal tree layout (preliminary version,",* STOC '80 Proceedings of the twelfth annual ACM symposium on Theory of computing, ACM New York, NY, USA, pp. 177–189.
[54] S. Chatterjee, et al., (2007) "*A linear time algorithm for optimum tree placement*", *Proceedings of International Workshop on Logic and Synthesis*, San Diego, California, USA
[55] M. Yannakakis, "*A polynomial algorithm for the min-cut linear arrangement of trees", Journal of ACM*, vol. 32, no. 4, pp. 950–988, 1985.
[56] Andrew Caldwell, et al., (1999)*, "Generic Hypergraph Formats, rev. 1.1",* from http://vlsicad.ucsd.edu/GSRC/bookshelf/Slots/Fundamental/HGraph/HGraph1.1.html.
[57] Jason Cong, et al, (2007), "*UCLA Optimality Study Project*", from http://cadlab.cs.ucla.edu/~pubbench/.
[58] Cadence®, (2006), "*Encounter® Menu Reference*", Product Version 6.1, Cadence® Design Systems, Inc., San Jose, CA, USA.
**Encounter*® is the trademark of Cadence Design Systems, Inc., San Jose, CA,USA.
[59] Saurabh Adya & Igor Markov, (2005), "*Executable Placement Utilities*" from http://vlsicad.eecs.umich.edu/BK/PlaceUtils/bin.
[60] Saurabh N. Adya, et al., (2003), "*Benchmarking For Large-scale Placement and Beyond*", *International Symposium on Physical Design (ISPD)*, Monterey, CA, pp. 95-103.
[61] Saurabh Adya and Igor Markov, (2002), "*Consistent Placement of Macro-Blocks using Floorplanning and Standard-Cell Placement*", *International Symposium of Physical Design (ISPD)*, San Diego, pp.12-17.
[62] Cadence®, (2004), "*LEF/DEF Language Reference*", Product Version 5.6, Cadence® Design Systems, Inc., San Jose, CA, USA.

AUTHORS

Santeppa Kambham obtained B.Tech. in Electronics and Communication engineering from J N T U and M Sc (Engg) in Computer Science and Automation (CSA) from Indian Institute of Science, Bangalore. He worked in Vikram Sarabhai Space Centre, Trivandrum from 1982 to 1988 in the field of microprocessor based real-time computer design. From 1988 onwards, he has been working in the field of VLSI design at ANURAG, Hyderabad. He received DRDO Technology Award in 1996, National Science Day Award in 2001 and "Scientist of the Year Award" in 2002. He is a Fellow of IETE and a Member of IMAPS and ASI. A patent has been granted to him for the invention of a floating point processor device for high speed floating point arithmetic operations in April 2002.

Siva Rama Krishna Prasad Kolli received B.Sc. degree from Andhra University, DMIT in electronics from MIT, M.Tech. in Electronics and Instrumentation from Regional Engineering College, Warangal and PhD from Indian Institute of Technology, Bombay. He is currently working as Professor at Electronics and Communication Engineering Department, National Institute of Technology, Warangal. His research interests include analog and mixed signal IC design, biomedical signal processing and image processing.