Interleaving of Gate Sizing and Constructive Placement for Predictable Performance

Sungjae Kim[1], Eugene Shragowitz[1], George Karypis[1], and Rung-Bin Lin[2]
[1]Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455
[2]Department of Computer Engineering and Science, Yuan-Ze University, Chung-Li, Taiwan
Email: {sukim, shragowi, karypis}@cs.umn.edu, csrlin@cs.yzu.edu.tw

***Abstract*— This paper presents a fast fixed-die standard cell placement algorithm. Placement is achieved by a combination of top-down partitioning with the incremental row-by-row construction. This paper concentrates on the construction part of this process. Gate sizing is interleaved with the placement construction process. Before placement, every gate is given its minimal size. During the placement, gates are resized to satisfy the timing constraints. Behavior of the placement is adapted based on dynamically recomputed net delay bounds. Experimental results show significant improvement in timing, predictability of results, and run time with respect to a commercial placement tool.**

I. Introduction

Circuit speed and power consumption always were and continue to be central to the switching circuit design. Now, as before, achieving projected timing closure and/or satisfying limits on power consumption is a major design goal. One of the major problem of contemporary design is a gap between requirements and outcomes of design steps. For example, it is common to see that after synthesis, placement, and routing by the most advanced commercial tools, the resulted delay on the longest path is 20-30% above the required value. The major cause of it is a physical design. Huge dimensions and NP-hard nature of the placement problem and difficulties in embedding timing and power constraints in physical design algorithms lead to poor timing performance of placed circuits.

Large number of timing-driven placement algorithms can be classified into two categories [1]: path-based and net-based approaches. Path-based approaches suffer from excessive computing time because of the exponential number of path to be optimized. With the increase of circuit sizes, net-based approaches become more attractive. In net-based approaches, requirements are converted into wire length constraints for individual nets [2], [3], [4]. One of the problems is how to overcome problems with physical placement feasibility. This issue is a main subject of this paper.

The rest of the paper is organized as follows: In section II, the modified data flow of CP (Constructive Placer) with a new gate sizing algorithm is provided. The gate sizing algorithm is presented in section III. Experimental results are given in section IV, followed by conclusion in section V.

II. Constructive Placement

Fig. 1 illustrates data flow of CP with a new gate sizing algorithm.

This algorithm starts from giving every cell its minimal size before placement. Delay bounds and net criticalities, which guide the placement construction process, are computed for initial gate sizing. With respect to the computed net criticalities, a circuit is partitioned and mapped to physical bins. Unlike partition-based placers, CP produces coarse partitions for faster run time and flexibility in the construction process. hMetis [9], a cutting-edge hypergraph partitioner, is integrated in CP. After that, the gate sizing algorithm is applied based on the initial delay bounds and net criticalities. Placement is constructed row-by-row and new delay bounds and criticalities are recomputed based on information from the partial placement. The new data are used for selection of new gate sizes. The whole process is repeated until all gates are placed. After finishing placement, orientations of cells are determined and locations of I/Os are assigned.

A. Delay Bounds and Net Criticality Computation

Net delay bounds play an important role in CP. CP employs a modified IMP algorithm to compute net delay bounds. The early version of IMP algorithm was proposed in [5]. The algorithm has unique properties not demonstrated by other zero slack algorithms [7], [8]. IMP algorithm computes bounds on the nets, based on the full topology of the circuit and uses the net physical characteristics (pin number, driver size, or others) as weights.

If S_p is the slack at the primary output for the path p, it is distributed between the nets on this path according to the weight of nets $w(e)$. Thus, based exclusively on single path p the delay bound could be computed as

$$Bound_p(e) = S_p \times \frac{w(e)}{\sum_{e' \in p} w(e')} \quad (1)$$

However, the net e is traversed by multiple paths. Therefore, a bound should be selected as a minimum of all maximal bounds on delay of the net e computed for all individual paths traversing this net.

$$Bound(e) = min\{max_{p \in \Pi_e} Bound_p(e)\} \quad (2)$$

where, Π_e is a set of paths, that traverses net e. The problem (2) is NP-hard. Basic idea of IMP algorithm is very simple. Rewriting formula (1) gives $Bound(e) = w(e) \times \frac{S_p}{\sum_{e' \in p} w(e')}$,where $w(e)$ is a constant and minimum of a

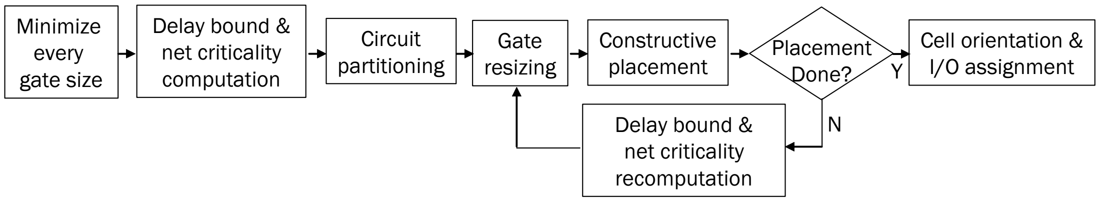
Minimize every gate size
Delay bound & net criticality computation
Circuit partitioning
Gate resizing
Constructive placement
Placement Done?
Y
Cell orientation & I/O assignment
N
Delay bound & net criticality recomputation

Fig. 1. Dataflow of the proposed algorithm.

ratio $S_p / \sum_{e' \in p} w(e')$ is defined on the path p. The lower bound on this ratio could be obtained if $S_p^{min} \leq S_p$ and defined as $S_p^{min} = min_{p \in \Pi_e} S_p$ i.e. as a path with the minimal slack at the output among all the paths traversing net e, and $W(e)^{max} \geq max \sum_{e' \in p}(w(e'))$, i.e. as a weight of the path with the maximal weight among those traversing net e.

We modified IMP algorithm based on the fact that a node slack is equal to the path slack of the longest path traversing the node. In our work, S_p^{min} is replaced by a node slack improving the run time of IMP algorithm.

Net criticalities are computed based on net delay bounds and characteristics of nets. In CP, the net criticality is defined as in [6]:

$$Net\ Criticality = \frac{Fanout\ of\ Net}{Net\ Delay\ Bound} \tag{3}$$

B. Constructive Placement

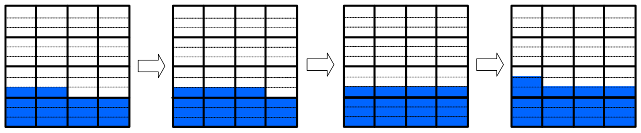

Fig. 2. Constructive placement process. Dark region is a partial placement. Partitions are represented by thick solid lines.

Fig. 2 illustrates the constructive placement procedure. The placer produces the legal placement incrementally row-by-row. Inside a row, construction is performed for each partition separately. Non-placed cells from a partition are considered as candidates for placement. Candidate cells are selected based on the cell criticality which is defined as the sum of net criticality $\times$ connectivity to the partial placement of all incident nets to the candidate. The candidate cells are sorted in decreasing order of cell criticality and according to that order, cells are placed in their feasible regions. The feasible region for a candidate is the region in a row where the sum of the horizontal components of connections to already placed incident cells is minimal. If the feasible region for a candidate is already occupied by the other candidate cells with the higher cell criticality values, then the placement of the candidate is deferred to the next row. However, if the vertical connection increment due to deferring the candidate to the next row is greater than the horizontal connection length increment, then the candidate is still selected for placement in the row and its feasible region is extended. The optimal location with the minimal length of interconnections for each candidate is identified by the exhaustive search of possible locations in the feasible region.

Because of the differences between projected and realized delay on placed nets, the net delay bounds are recomputed after placement of the set of rows. Actual HPWL (Half Perimeter Wire Length) of the nets can be computed for a constructed part. Resistance and capacitance of the nets are estimated from the wire load table of the standard cell library based on the HPWL. The estimated resistance and capacitance are used to compute the delay of the net using a RC delay model. The new delay bounds for non-placed nets are computed by the same delay bound computation algorithm.

III. Gate Sizing Algorithm

It is clear that timing can be improved by using larger drivers on timing-critical nets. There are two questions to be answered for solving this problem: How to identify the minimal number of cells for up-sizing to achieve desirable timing and how to select cell sizes without violation of the space requirements? In CP algorithm, cells are resized in time of a slice construction. Net criticalities derived from recomputated delay bounds are used for ranking nets' drivers in descending order of their criticalities. In our experiments, this approach turned out to be more effective than on described in [10].

The new gate sizing algorithm is presented in Algorithm 1.

Non-placed cells are sorted in decreasing order of criticalities of their output nets. For each non-placed cell, the remaining space of the partition, to which the cell belongs, is computed. To maximize potential delay reduction of non-placed cells, functionally equivalent cells are considered in the decreasing order of their sizes. Starting from the largest equivalent cell, available space in the partition is compared with the size of the equivalent cell. If there is enough space, then it is selected for placement. After resizing, delay information and available space of the partition is updated. And a new row utilization factors are generated based on the size increment of the cell and remaining space of partitions.

IV. Experimental Results

CP was implemented in C language. All experiments were performed on Sun UltraSPARC machines with 1.5GHz CPU and 1GB memory. The benchmark suite consists of 16 large circuits from the ITC'99 [11] benchmark set. Netlists were converted into the Verilog format using the Synopsis edif2verilog converter. Each circuit was optimized by the Cadence BuildGate synthesis tool for values of provided

Algorithm 1 Gate sizing algorithm.

```
1: · $EQSET_C$ : set of functionally equivalent cells for cell C. $EQSET_C$ is pre-sorted according to decreasing order of cell sizes;
2: · $NPC$ : set of non-placed cells;
3: $l$ = number of cells in $NPC$;
4: Minimize the sizes of cells in $NPC$;
5: Sort $NPC$ according to the output net criticalities;
6: for ($i = 0; i \le l; i$++) do
7:    $C = NPC[i]$;
8:    $P$ = the partition for $C$;
9:    $CAP$ = the remaining space in $P$;
10:   $k$ = number of cells in $EQSET_C$;
11:   for ($j = 0; j \le k; j$++) do
12:      $E = EQSET_C[j]$;
13:      if width of $E \le CAP$ then
14:         $C = E$; /*replace C by E */;
15:         goto line 18;
16:      end if
17:   end for
18:   if $C$ is replaced then
19:      Update delay information;
20:      Update $CAP$;
21:      Compute new utilization factor for $P$;
22:   end if
23: end for
```

timing constraints. The 0.18μm, 5-metal layer standard cell library from Virtual-Silicon Technology Inc. [12] was used in the experiments. For interconnect parameters and switching delays of cells, we used ITRS [13] data for year 2005.

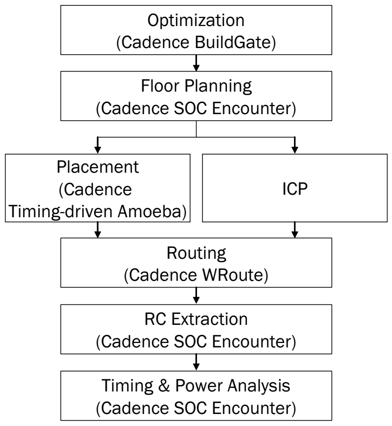

Fig. 3. Flow of experiments.

Organization of experiments is given in Fig. 3. Each circuit in the set was optimized for 4 different clock frequencies (increased by 10% per step). The optimization step provided the netlists with the sized gates. The floorplanning tool generated the area estimation and definitions of rows with utilization factor 0.85 for the all optimized netlists. These floorplans were provided to CP and AMOEBA placer with the optimized netlists. The placement solutions were routed by WRoute router. Routed solutions were sent for RC extraction for timing and power analysis to the Cadence SOC Encounter.

Because of the space limitations, we presented experimental results for all circuits in the list for one timing constraint (20% increase in clock frequency w/r to the initial frequency) in Table I. Columns 1-4 give circuits' names and basic statistics. The timing constraints used in optimization by the Candence BuildGate synthesis tool are shown in column 5. Columns 6 and 7 present delays reported by AMOEBA (AM) and CP, and column 8 gives the ratios. The average delay reduction is 21%. Columns 9, 10, and 11 report routed wire length and ratios. On the average, the solutions by CP for wire length are marginally better (by 1%), i.e. CP provides the better timing solutions without an increase in the wire length. Run times of both placers are given in columns 12-14. CP is 2.66 times faster than Amoeba.

Table II shows the requested and achieved delay of the solutions from CP and AMOEBA for the circuit b22 with 4 different timing constraints.

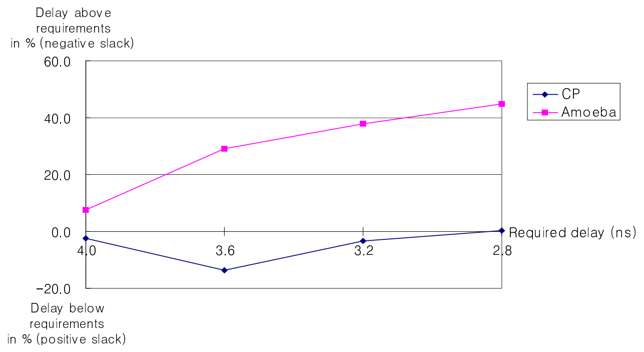

Fig. 4. Deviation of achieved delay from required for b22 by two placers.

Fig. 4 presents deviation of the actual delay from the requirements for b22 in Table II. The deviation is calculated as 100$\times$(actual delay - required delay)/required delay. A negative deviation indicates that the requirement is satisfied and a positive deviation means that the design solution has negative slack. It is easy to see that the 3 delay constraints out of 4 were satisfied by CP solutions while AMOEBA solutions violated requirements in all 4 cases. The average deviation of the actual delay above the required delay is 0.05% (0.2/4) for CP, and for AMOEBA, deviation is 29.9% ((7.7+29.0+37.9+44.9)/4).

Fig. 5 illustrates the generalization of Fig. 4 for all of the designs and all frequencies. The average deviation of delay above requirements is 2.2% from CP. For the AMOEBA placer, deviation is 17.9%.

TABLE III
DELAY RATIOS AND % OF SATISFIED TIMING REQUIREMENT.

Required delay reduction ratio		1.0	0.9	0.8	0.7	Avg.
Average delay ratio ($\frac{CP}{AM}$)		0.81	0.82	0.79	0.80	0.81
% of satisfied timing requirement	CP	79	64	75	36	63
	AM	57	29	19	0	26

Table III summarizes delay ratios between solutions of two placers and the % of satisfied timing requirement for all timing constraints. On the average CP produces solutions with 19%

TABLE I

EXPERIMENTAL RESULTS ON DELAY, WIRE LENGTH, AND RUN TIME FOR ONE SET OF TIMING CONSTRAINT (20% REDUCTION IN REQUIRED DELAY).

Ckts	#Cells	#Nets	#Rows	Required	Delay (ns)			Wire length (μm)			Run Time (sec)		
				Delay	AM	CP	$\frac{CP}{AM}$	AM	CP	$\frac{CP}{AM}$	AM	CP	Speed Up
b14_1	5027	5061	50	2.4	2.966	2.001	0.67	159651	141928	0.89	17	7	2.43
b15	8328	8366	65	1.6	3.623	2.842	0.78	316074	320266	1.01	43	17	2.53
b14	5771	5805	53	2.7	3.144	2.186	0.70	175543	182471	1.04	14	12	1.17
b15_1	8261	8299	65	1.6	3.120	2.420	0.78	291558	297305	1.02	38	14	2.71
b20_1	11182	11216	73	2.7	3.411	2.333	0.68	340831	309009	0.91	58	23	2.52
b21_1	10837	10871	73	3.2	3.891	2.796	0.72	328013	316038	0.96	50	23	2.17
b20	12063	12097	76	3.2	4.110	2.856	0.69	383922	367134	0.96	53	25	2.12
b21	12491	12525	77	3.2	3.728	2.776	0.74	388501	419290	1.08	54	17	3.18
b22_1	16323	16357	89	3.2	4.155	3.480	0.84	513596	531373	1.03	102	35	2.91
b22	18264	18298	93	3.2	4.412	3.091	0.70	583867	582463	1.00	107	36	2.97
b17	26069	26108	116	3.2	4.382	3.702	0.84	1026510	1040100	1.01	132	57	2.32
b17_1	24973	25012	114	2.4	2.911	2.296	0.79	943508	969119	1.03	128	61	2.10
b18_1	79131	79171	197	4.0	3.824	3.728	0.97	2664992	2615974	0.98	998	220	4.54
b18	80668	80706	198	4.0	3.906	3.689	0.94	2699992	2648570	0.98	1078	234	4.61
b19_1	143579	143631	266	4.0	4.332	3.832	0.88	4924101	4974201	1.01	1482	692	2.14
b19	146347	146399	268	4.0	3.877	3.429	0.88	4999159	4977246	1.00	1497	721	2.08
Avg.							0.79			0.99			2.66

TABLE II

REQUESTED AND ACHIEVED DELAY FOR B22.

	Req. Delay(ns)	4.0			3.6			3.2			2.8			Avg.
b22	Achieved	AM	CP	$\frac{CP}{AM}$	AM	CP	$\frac{CP}{AM}$	AM	CP	$\frac{CP}{AM}$	AM	CP	$\frac{CP}{AM}$	
	Delay(ns)	4.306	3.897	0.91	4.643	3.104	0.67	4.412	3.091	0.70	4.056	2.806	0.69	0.74

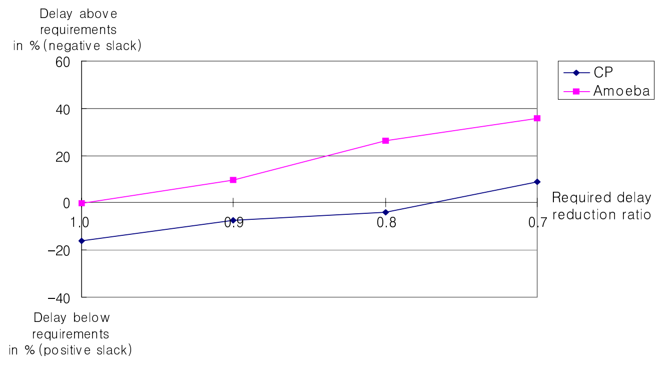

Fig. 5. Average deviation of achieved delay from required for all test cases and all frequencies.

better timing. The % of satisfied timing requirement was calculated by (#of circuits which satisfy timing constraint / total number of circuits in the experiments). CP satisfies timing constraints in 63% of cases on the average, while AMOEBA satisfies the constraints in 26% of cases.

V. CONCLUSION

Interleaving of constructive placement and gate sizing is demonstrated to be an effective approach in reducing delay (19% on the average) and increasing predictability of outcomes in the design process by more than 2 times. The success of this approach depends on dynamic gate resizing accompanied by recomputation of delay bounds by the modified IMP algorithm in conjunction with a design partitioning procedure and constructive placement. The placer is 2.66 times faster than AMOEBA.

REFERENCES

[1] J. Cong, T. Kong, J. R. Shinnerl, M. Xie, and X. Yuan, ”Large-Scale Circuit Placement: Gap and Promise,” *Proc. of ICCAD* , page 883-890, 2003.

[2] J. Y. Sayah et. al., ”Design planning for high-performance ASICs,” in *IBM Journal of Research and Development*, Vol. 40, No. 4, pp. 431-452, 1996

[3] X. Yang, B. Choi, and M. Sarrafzadeh, ”Timing-Driven Placement using Design Hierarchy Guided Constraint Generation,” *Proc. of ICCAD*, pp. 177-180, 2002

[4] S. Kim and E. Shragowitz, “Iterative-Constructive Standard Cell Placer for High Speed and Low Power,” *Proc. of ICCD*, pp. 350-355, 2006.

[5] H. Youssef, R-B Lin, and E. Shragowitz, “Bounds on Net Delays for VLSI Circuits,” *IEEE Transactions on Circuits and Systems*, Vol. 39, No. 11, pp. 815-824, 1992.

[6] H. Chang, E. Shragowitz, J. Liu, H. Youssef, B. Lu, and S. Sutanthavibul, “Net Criticality Revisited: An Effective Method to Improve Timing in Physical Design,” *Proc. of ISPD*, pp. 155-160, 2002.

[7] R. Nair, C. L. Berman, P. S. Hauge, and E. J. Yoffa, ”Generation of Performance Constraints for Layout,” In *IEEE Transactions on CAD*, Vol. 8, No. 8, pages 860-874, 1989.

[8] C. Chen, X. Yang, and M. Sarrafzadeh, “Potential Slack: An Effective Metric of Combinational Circuit Performance,” *Proc. of ICCAD*, pp. 198-201, 2000.

[9] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, ”Multilevel Hypergraph Partitioning: Application in VLSI Design,” *Proc. of DAC*, pages 526-529, 1997.

[10] J. P. Fishburn and A. E. Dunlop, ”TILOS: A Posynomial Programming Approach to Transistor Sizing,” *Proc. of ICCAD*, page 326-328, 1985.

[11] ITC99 Benchmarks, http://www.cerc.utexas.edu/itc99-benchmarks/bench.html.

[12] Virtual-Silicon Technology Inc., http://www.virtual-silicon.com.

[13] International Technology Roadmap for Semiconductors, http://www.itrs.net.