A PROBABILISTIC APPROACH TO BUFFER INSERTION

*Vishal Khandelwal, Azadeh Davoodi, Akash Nanavati * and Ankur Srivastava*

Department of Electrical and Computer Engineering, University of Maryland at College Park
* Computer Science Department, University of California at Los Angeles
vishalk,azade,ankurs@glue.umd.edu, * akash@cs.ucla.edu

Abstract

This work presents a formal probabilistic approach for solving optimization problems in design automation. Prediction accuracy is very low especially at high levels of design flow. This can be attributed mainly to unawareness of low level layout information and variability in fabrication process. Hence a traditional deterministic design automation approach where each cost function is represented as a fixed value becomes obsolete. A new approach is gainng attention [15, 5, 2, 4, 12] in which the cost functions are represented as probability distributions and the optimization criteria is probabilistic too. This design optimization philosophy is demonstrated through the classic buffer insertion problem [13]. Formally, we capture wirelengths as probability distributions (as compared to the traditional approach which considers wirelength as fixed values) and present several strategies for optimizing the probabilistic criteria. During the course of this work many problems are proved to be NP-Complete. Comparisons are made with the Van-Ginneken "optimal under fixed wire-length" algorithm. Results show that the Van-Ginneken approach generated delay distributions at the root of the fanout wiring tree which had large probability (0.91 in the worst case and 0.55 on average) of violating the delay constraint. Our algorithms could achieve 100% *probability of satisfying the delay constraint with similar buffer penalty. Although this work considers wirelength prediction inaccuracies, our probabilistic strategy could be extended trivially to consider fabrication variability in wire parasitics.*

1. INTRODUCTION

Design automation of integrated systems is essentially optimization driven by estimation. If the estimation of critical design objectives is inaccurate, the optimality of optimization will be limited. Unfortunately, estimation is always marred with inaccuracies which occur due to many factors. Unawareness of exact implementation information, unpredictable circuit behavior, fabriaction variability are important ones among them. Traditionally, optimization in design automation has been deterministic since it assumes a fixed value to the pertinent cost function (like area, delay, power). Lately, a new optimization approach is gaining attention [15, 5, 2, 4, 12] in which the cost functions are represented as probability distributions and the optimization criteria is probabilistic too. Such a design methodology would be able to address the issue of prediction uncertainities and fabrication variability in a much more robust fashion when compared with traditional deterministic approach. In this paper we present such an optimization methodology for the buffer insertion problem [13]. We address unpredictabilities posed by wirelength estimation and/or variation of interconnect properties due to fabrication variability [15] and illustrate the superiority of our probabilistic approach over traditional deterministic Van-Ginneken algorithm [13].

The problem of buffer insertion deals with the placement of buffers at appropriate positions on the fanout wiring trees such that the delay at the driving gate is minimal. Lukas van Ginneken [13] presented an optimal buffer placement algorithm for RC-Trees under the Elmore Delay model for wires. It was assumed that wirelengths of individual segments in the wiring tree are known a-priori through some estimation engine. Estimating wirelengths especially in a traditional top down design flow is extremely hard and error prone. This makes estimation of delay and capacitive loading of the individual wire segments extremely difficult. Even if the wirelength estimates are accurate, the fabrication variability/uncertainty makes accurate estimation of wire parasitics an intractable problem. Hence, the traditional deterministic approach possesses serious distadvantages. In this paper we extend Van-Ginneken's approach to consider wirelength estimation inaccurcay by modeling it as probability distributions. We propose a new probabilistic criteria of selecting the final solution. The new criteria computes and minimizes the probability of violating a given delay constraint. The paper also presents three algorithms for performing buffer insertion when wire-lengths are assumed as distributions. These three approaches have different pruning criteria (from very relaxed to very strict) and varying runtime complexities. Several sub-problems were proved NP-Complete indicating that finding an optimal solution in polynomial time under the probabilistic wire-length assumption is very hard.

Experiments were conducted on large benchmarks with state of the art technology parameters. Comparisons were made with the Van-Ginneken approach [13]. Results showed that the Van-Ginneken approach generated delay distributions at the root of the wiring tree which had large probability (0.91 in the worst case and 0.55 on average) of violating the delay constraint. Our algorithms could achieve 100% probability of satisfying the delay constraint with similar buffer penalty. This is a very strong result since our approach ensures that the delay constraint will always be satisfied.

Although in our approach we address wire-length predication inaccuracies, our algorithms can be trivally extended to the case where the estimation of wire parasitics is also inaccurate due to fabrication variability.

The rest of the paper is organized as follows. In section 2, we formally state the buffer insertion problem and present the traditional

approach [13]. Section 3 presents the selection criteria in a probabilistic scenario, and section 4 presents our Probabilistic Buffer Insertion Algorithms with different pruning criteria. We discuss our results in section 5. The paper is concluded in section 6.

2. MOTIVATION

2.1. The Probabilistic Paradigm

Automation of integrated systems is marred with estimation inaccuracies which occur due to a combination of many factors. Unawareness of exact layout information like routing, placement, exact logic structure are prominent reasons. Lately fabrication uncertainties have also begun to get considerable weight primarily due to increasing complexity and scaling of the fabrication process. In the light of such unpredictabilities, a traditional deterministic approach towards design automation becomes incapable and obsolete. Basically, a deterministic approach assigns a fixed value to the cost function (like area, delay, power) and does not consider the error associated with the estimation of this cost function. Hence, very little could be said about the optimality of the final design especially if the estimation was erroneous. This calls for the development of a probabilistic approach towards design optimization. Such an approach models the cost functions as probability distributions and optimizes the design probabilistically, hence maximizing the likelihood of satisfying design constraints. Many researchers have suggested the importance of such an approach [1, 5, 15, 4, 12, 2] since estimation inaccuracies (both due to fabrication variability and layout unawareness) are becoming major bottlenecks in design closure. The main advantage of such an approach would be faster design closure, better fabrication yield (since fabrication variability would have been accounted for during designing) and improved robustness.

In this paper we present such a probabilistic approach for the classic buffer insertion problem. We revisit the traditional deterministic buffer insertion approach proposed by Van-Ginneken and reformulate the problem probabilistically. In this work we assume the source of unpredictability to be the inaccuracy in wirelength estimation. Hence we model the wirelengths as probability distributions. Our approach can be trivially extended to the case when the paramemters of the wires (and not the length itself) change due to fabrication variability.

2.2. Traditional Buffer Insertion

2.2.1. The Problem

The Buffer Insertion Problem can be formally stated as:

Given the fanout wiring tree with parasitic resistances and capacitances, wire-lengths, potential buffer locations, sink required times, sink capacitive loads and a delay constraint at the driving gate, the problem is to place buffers into the tree such that the required arrival time at the input of the driving gate is maximum. We also consider the optimization of the number of buffers used to satisfy the delay constraint.

The buffer insertion problem formalized by [13] models the fanout wiring tree as a set of distributed RC sections. The Elmore Delay model [16] is used to compute the delay of such a wiring tree. Figure 1 illustrates a typical wiring tree. Each of the individual wire segments is characterized by parasitic resistances and capacitances

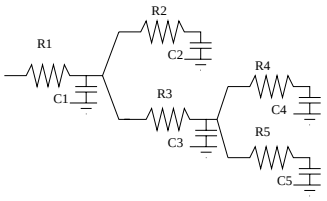


Figure 1: RC Tree Network

(like R3 and C3). These depend on the length of the corresponding wire. A subtree rooted at node k is represented by two numbers: the required arrival time T_k and capacitive loading L_k.

Adding a wire of length l at the root of a subtree affects the T_k,L_k values as :

$$T'_k = T_k - rlL_k - (1/2)rcl^2 \quad (1)$$

$$L'_k = L_k + cl \quad (2)$$

When a buffer is added at the root of the subtree :

$$T'_k = T_k - D_{buf} - R_{buf}L_k \quad (3)$$

$$L'_k = C_{buf} \quad (4)$$

When two subtrees rooted at n and m are merged into one subtree :

$$T'_k = min(T_n, T_m) \quad (5)$$

$$L'_k = L_n + L_m \quad (6)$$

These equations can be used to compute the required arrival time T_o at the root of the wiring tree. For brevity we have omitted the detailed description of this delay model. This is a very popular delay model for capturing wire-delays in modern layout driven optimization systems. A lot of research has been done on the buffer insertion problem [10, 17, 8, 3, 11] which is especially useful for large global nets like clock trees. Most of these approaches use a dynamic programming based approach in which the wiring tree is traversed topologically from sinks to source while storing an optimal solution set. Next we describe Van-Ginneken approach to buffer insertion which solves the problem optimally for a fixed wiring topology and buffer placement locations.

2.2.2. The Van-Ginneken Algorithm

Using the delay model described above, Van-Ginneken proposed his buffer placement algorithm [13]. The input to his strategy was a wiring tree with estimated parasitics and a set of possible buffer locations, fanout capacitive loadings and required arrival times. The wire parasitics in turn are dependent on the wire-lengths which are assumed to be prespecified. The problem is to decide buffers locations at the prespecified positions such that the required time at the root is maximized. The Van-Ginneken strategy is optimal under the Elmore Delay model. His algorithm traverses the wiring tree topologically from primary outputs to primary inputs. At each possible buffer location, it evaluates the possibility of adding a buffer and its effect on the required time and capacitive loading at that node. This is followed by local pruning of the generated solutions. The pruning criteria removes those solutions from the set of potential solutions which have another solution with better values of both required time and capacitive loading. Lukas Van-Ginneken[13] proved that this

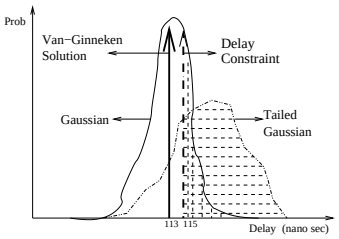


Figure 2: Mean Value vs. Actual Delay Distribution

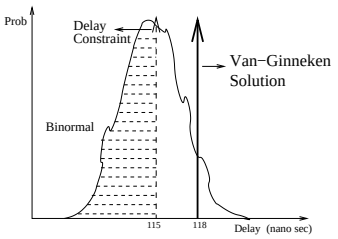


Figure 3: Worst Case Length Estimate

pruning criteria generates an optimal solution at the root in polynomial time. Once again for brevity we do not go into the details of this algorithm.

2.3. Shortcoming of Existing Approach

Existing approaches to Buffer Insertion (or any other problem in design automation) do not consider the uncertainties in estimating wire-lengths (or any other pertinent cost function). Focusing the discussion on Buffer Insertion, the Van-Ginneken approach assumes wire-lengths to be prespecified from some estimation engine as fixed values. In reality, no estimation engine can give accurate wire-length predictions. This is due to the unawareness of future optimizations and hence the state of the final design. This is also due to the sensitivity between various cost functions. For example, if congestion in a certain region is intolerable, then any strategy of optimizing congestion could have adverse effects on wire-length. Using a fixed value of wire-length therefore cannot capture the real variations involved.

In this work we relax the assumption of having fixed wire-length estimates and propose a new buffer insertion strategy which probabilistically models lengths and optimizes the underlying distribution. But before we delve into the details of our algorithm, we quantitatively illustrate the shortcomings in making a fixed wire-length estimate.

Let us assume that we know the distribution of length of the wire-segments. The Van-Ginneken algorithm does not consider distributions, hence we consider the two ways of providing fixed length values to the algorithm:

1. Average length of the distribution
2. Worst (longest) length in the distribution

We conducted experiments with these two wire-length estimates. The Van-Ginneken algorithm was given these estimates and the buffered solution was generated. On this buffered tree, we imposed the real distribution of wire-lengths. Using these distributions, the delay distribution at the root was computed.

Figure 2 illustrates the results when the average values from the wire length distribution were given as wire-length estimates. The figure reports the variation in delay at the root for the result generated by the Van-Ginneken algorithm. These plots are shown for different wirelength statistical distributions (tailed gaussian and gaussian). The bold arrow illustrates the delay estimated by the Van-Ginneken algorithm. In reality the delay values at the root are distributions. It can be seen that there is a large portion of these distributions whose delay is greater than that estimated by the Van-Ginneken algorithm. Let us suppose we have a delay constraint illustrated by the dotted line. According to the Van-Ginneken solution this constraint is satisfied, but in reality, there is a large portion of the delay distribution that violates this constraint. This clearly shows that the deterministic fixed wirelength approach can result in failure of design closure.

We also assigned the worst case length values as estimates. Figure 3 presents the data for gaussian wire-length distributions. Such an estimation strategy could be an overestimate and hence an overkill. It can be seen that for the same delay constraint as in figure 2, the Van-Ginneken solution (shown in a bold arrow) will not be able to satisfy the constraint. But in reality (by observing the delay distribution of the Van-Ginneken buffered tree), we find that there is a large area which lies within the constraint (the shaded area). Hence the Van-Ginneken result would be an overkill both in terms of number of buffers and delay.

What we observed so far was that using fixed values to estimate cost functions does not accurately address the issue of unpredictabilities. In reality, these cost functions should be modeled as distributions and the algorithms should be re-formulated to consider these distributions. In this work we present such an approach for the buffer insertion problem.

3. PROBABILISTIC BUFFER INSERTION: METRICS

The previous section illustrated the importance of considering probability distributions of cost functions during optimization. Let us consider the following situation in this light. Given a wiring tree with possible buffer locations and a delay constraint at the root, the problem is to place buffers such that the delay constraint is satisfied. Consider two given solutions at the root, each corresponding to different distributions (as illustrated in figure 4). The figure also illustrates the delay constraint that needs to be satisfied. Here the distribution with larger spread has lesser mean value for delay compared with the distribution with smaller spread. A traditional approach would choose the distribution with smaller mean (see figure 4). Clearly this solution has a large area outside the delay constraint. Hence it has a larger probability of failure. In this situation the second solution with a smaller spread should be the choice. This observation can be formally outlined as follows

$$Minimize \sum_{d \geq D_{cons}} p(d) \tag{7}$$

Here
d : Delay p(d): Probability of delay d

Basically, we would like to choose a solution that minimizes the total probability of the delay constraint D_{cons} not being satisfied.

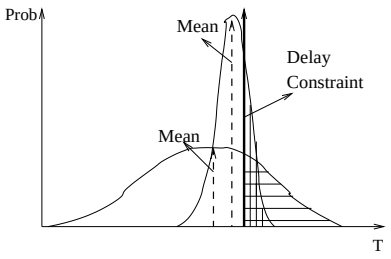


Figure 4: Spread in Distribution

This is a probabilistic selection criteria.

4. PROBABILISTIC BUFFER INSERTION: ALGORITHMS

In the next few sections we will describe algorithms to optimize the criteria outlined above. The input to our algorithms is a wiring tree with parasitic resistances and capacitances, distributions of wire-lengths (instead of fixed values), possible buffer locations, sink required times, sink capacitive loads and a delay constraint at the root. The root is assumed to be a Nand gate which drives the wiring tree. The delay constraint needs to be satisfied at the input of this gate. We modify the Van-Ginneken approach to consider probability distributions of wire-lengths, the details are outlined below.

4.1. The Global Algorithm

The **Global Algorithm** approaches the problem similar to the Van-Ginneken strategy. The RC-Tree network is traversed topologically from sinks towards source. At the root of each subtree (internal node in the network), the set of possible solutions are computed by merging the solutions of fanout-subtrees. Just like the Van-Ginneken approach each solution comprises of two entities: T and L. T corresponds to the required arrival time at the root and L corresponds to the capacitive loading. The difference lies in the fact that since wire-lengths are considered as distributions, both T and L will be distributions too. Refering to figure 5(b), at any internal node i we have the option of whether to place a buffer or not. In order to compute the potential solutions at i, the solutions at its fanout nodes j and k are propagated to i (by adding the delay distribution of the wires (x and y which connect them to i respectively). This can be done using equations 1 and 2 (although all variables are distributions now). Now we generate the possible solutions at node i by merging these modified solutions from the fanout nodes using equations 5 and 6. If node i has two fanouts with m and n solutions each, then there could be a total of mn solutions at i. The possibility of buffer placement at node i, makes this solution set to become $2mn$. Each buffer is characterized by an input capacitance C_{buf}, and internal delay D_{buf} and an output impedance R_{buf}. Adding a buffer modifies the solutions at node i according to equations 3 and 4. All these equations would be applied on probability distributions of L and T instead of fixed values.

The total solutions at node i can become very large (non-polynomial in problem size). Hence this solution set needs to be pruned. Van-Ginneken has a very effective pruning criteria in which both polynomiality and optimality were achieved [13]. Since the L and T values are not fixed, the Van-Ginneken pruning criteria cannot be used. We propose three probabilistic pruning strategies with varying complexities in the next few sub-sections. After pruning, we are left with a reduced set of solutions at node i. A dynamic programming implementation computes the delay distribution of all potential solutions at the input of the gate driving the RC-Tree. The solution which has the highest likelihood of meeting the delay constraint is chosen as the final solution of the buffering problem(using equation 7).

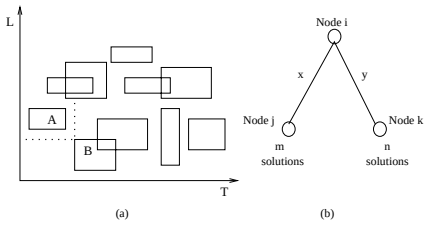


Figure 5: Distribution of Potential Solutions at a node

4.2. Pruning Strategies

We now propose three different pruning strategies for the Global Algorithm:

4.2.1. Criteria 1

Let us assume that at a given sub-tree we have all the possible solutions (including the possibility of adding a buffer at the root of subtree). Let us make the following observation. A specific solution i at the root k of a subtree is actually a distribution in required time (T_k^i) and a distribution in capacitive loading L_k^i. On a two dimensional plane with x-axis as T and y-axis as L, this solution can be represented as a rectangle in the worst case. The length of the rectangle is bounded by the smallest and largest required time values in the corresponding distribution. Similarly, the width of the rectangle is defined by the range of capacitive loading. For any point (x,y) inside the rectangle (x being the required time axis, y being the loading axis), p(x)p(y) denotes the probability of having x as the required time and y as the loading. Note that the probability that a solution point lies outside this rectangle is zero for the corresponding solution. Also note that if this was a deterministic approach, each solution would be represented as a point instead of a rectangle, which is exactly the case in the Van-Ginneken algorithm [13]. Figure 5(a) illustrates this concept and shows the distribution of the possible ssolutions at a node k. Consider two potential solutions A and B as marked in figure 5(a) for node k. It can be seen that B is better than A both in terms of the required time and capacitive load distributions. The dotted lines show that A does not even partially overlap with B along both T and L axis. Hence we can conclude that solution A is guaranteed to be worse than B and can be pruned out.

Formally the steps can be outlined as follows :

1. Given a root k of a subtree with a set of possible solutions (T,L) represented as rectangles
2. Prune out a rectangle which is definitely worse in capacitive loading and required time than another solution
3. The remaining set of solutions are considered co-optimal

Under this pruning strategy, the worst case number of solutions at the root can be exponential in the total possible buffer locations. The strategy also ensures that the optimal solution stays in the co-optimal set generated at the root.

4.2.2. **Criteria 2**

Once again, we assume that we have all possible solutions at a subtree k. In this approach, we have a stricter selection criteria which prunes out more potential solutions. We determine the probabilistic relation between a pair of potential solutions. Each pair (A,B) can have three possible relations: A can probabilistically prune out B, B can probabilistically prune out A or (A,B) could be co-optimal. These relations are elaborated as follows. First we would like to outline that given two distributions X and Y, probability that $X \geq Y$ is defined as $\sum_{y \in Y}(p(y) \sum_{x \in X, x \geq y} p(x))$. Given the distribution for T and L values for A and B, probabilities for the three possible relations is computed as follows.

$$A\ prunes\ B : P(A \Rightarrow B) = P_T(A \geq B) \cdot P_L(A \leq B) \quad (8)$$

$$B\ prunes\ A : P(B \Rightarrow A) = P_T(B \geq A) \cdot P_L(B \leq A) \quad (9)$$

$$Co-exist : P(A \sim B) = (1 - P(A \Rightarrow B) - P(B \Rightarrow A)) \quad (10)$$

Equation 8 computes the probability for A being better than B both w.r.t. required time ($P_T(A \geq B)$) and loading capacitance ($P_L(A \leq B)$). These values could be easily computed since we know the L and T distributions. Similarly equation 9 gives the probability of B being better than A. Equation 10 gives the probability of co-existence. The dominant relation between A and B is given by

$$max(P(A \Rightarrow B), P(B \Rightarrow A), P(A \sim B)) \quad (11)$$

Hence the relationship between two solutions A and B is decided by the one with highest probability. Now let us instantiate a graph G= (V,E) with each solution as node in G and edges defined as follows. If A prunes B according to equation 11 then there is a directed edge from node A to node B. If two nodes do not have any edges then they are co-existing solutions according to equation 11. Also note that there can be at-most one edge between any two nodes. We apply our pruning criteria on such a directed graph.

The main aim of pruning is to reduce the given set of solutions while maintaining the quality. The cost of a node δ_i is defined by the total number of nodes that can be pruned out by this solution. This essentially corresponds to the out-degree of this node. We want to find a maximum set of nodes such that:

1. All nodes in the set are co-optimal/independent
2. Each node not in this set has a directed edge from at-least one of the nodes in this set (note the keyword from)
3. The cost of the set $\sum_i \delta_i$ is maximum

This is a variation of the maximum independent set problem [9]. The variation is that each node not in the independent set must have an incoming edge from at-least one node in the independent set. The logic behind this constraint is that if a node (or a solution) is culled out, then there must be at-least one solution in the independent set that prunes this solution out (according to the probabilistic criteria described in the equations above). Moreover,

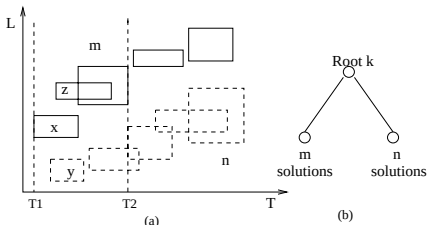


Figure 6: Generate Solutions at a Node from its Fanout Nodes

we want to maximize the total cost of all nodes, since the cost signifies the quality of a node (larger the cost, more are the number of solutions it prunes out). We name this problem as the DIRECTED MAXIMAL INDEPENDENT SET problem.

Theorem: DIRECTED MAXIMAL INDEPENDENT SET problem is NP-Complete.
Proof: Transformation from $Vertex - Cover$ [9]. Rest of the proof is omitted for brevity.

Heuristic for Criteria 2

Algorithm 1 *Heuristic for Criteria 2*

INPUT: Directed Graph G=(V,E);
Compute the cost of each node n = outdegree(n)
A: Set of all nodes
While(A != $\emptyset$)
    Choose highest cost vertex i from set A and add i to set B
    Remove all solutions from set A that have an edge with i
Return B

The heuristic used for this criteria is described in Algorithm 1. We sort all nodes w.r.t their cost values and iteratively pick the node with highest cost. The final solution generated by this algorithm will not satisfy the directed edge constraint of the Directed Maximal Independent Set problem. In fact the problem remains NP-C even if the directed edge constraint is relaxed since it is an instance of traditional Independent Set problem. Another point to note is that this is a stricter pruning criteria (prunes out more potential solutions) w.r.t. Criteria 1 but still does not guarantee a polynomial set of solutions at the root. Next we describe an even stricter approach which is completely polynomial.

4.2.3. **Criteria 3**

This approach is focussed on ensuring that the total number of solutions generated at the root of the wiring tree are polynomial in the possible buffer positions. Let us consider an internal node k with two subtrees (as shown in figure 6(b)). The two subtrees have m and n solutions respectively. In the worst case there will be $m \cdot n$ solutions at the root k after merging (assuming there is no buffer at the root). Merging two solutions from the left and the right subtrees essentially amounts to applying equations 5 and 6 to the distributions of the corresponding solutions. Let us suppose we are trying to merge solutions x and y in figure 6(a). The generated solution has

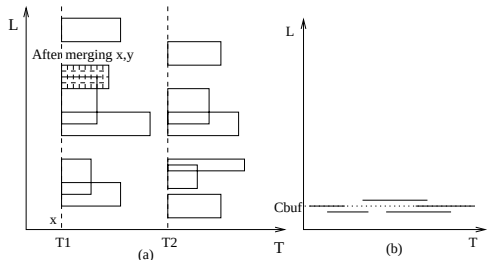


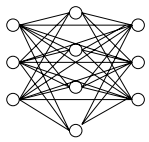
Figure 7: Total $m \cdot n$ Solutions After Merging

a delay distribution starting from T1 (see figure 7(a)). Also whenever x is merged with any solution whose starting T value is greater than T1, the generated solution will start from T1. Since there can be at most $m + n$ distinct starting times in figure 6, the generated $m \cdot n$ solutions at root k will be clustered in at most $m + n$ starting times. This is illustrated in figure 7(a).

Now comes our pruning criteria. For each distinct starting time value (shown in figure 7(a)), we pick exactly one solution and prune the rest. Hence we have at most $m + n$ solutions at root k. The exact strategy of picking these solutions will be discussed in the subsequent paragraphs. Before that, let us consider the situation where we can add a buffer at root k. Hence for each of the $m \cdot n$ solutions, we have the choice of adding a buffer. Therefore the total number of solutions become $2 \cdot mn$. Note that for the buffered solutions the capacitance is no longer a distribution.This is illustrated in figure 7(b) where the capacitance is a fixed value. According to this pruning criteria, we pick exactly one of these $m \cdot n$ buffered solutions at root k. The selection criteria used is as follows. We compute the probability that a buffered solution is better than another using equation 8. For each solution, we get the cumulative probability values. Finally we choose the solution that has the largest value. This solution is probabilistically better than all the other buffered solutions. Hence the total number of solutions that we store at a subtree is O(m+n+1) (m+n for solution with no buffers and one with the buffer). This is a polynomial quantity. Even the Van-Ginneken algorithm [13] was storing at most $m + n + 1$ solutions at a subtree. This results in a polynomial number of solutions at the root of the wiring tree. For brevity we omit the proof of polynomiality, the proof is similar to [13].

Theorem: Criteria 3 is polynomial in the problem size
Proof: Proof is similar to [13] and is omitted for brevity.

Now we go into the details of how $m + n$ solutions are chosen from $m \cdot n$ possible solutions (for the case where there are no buffers). As mentioned before there will be $m + n$ distinct starting T values (figure 7(a)). We choose exactly one solution from each distinct starting T value. This problem is modeled using *COMPLETE R-PARTITE MAX COST CLIQUE* problem [14]. The transformation is as follows. We instantiate a complete R-PARTITE graph G=(V,E) with the following properties. For each set of solutions that have the same starting T value, we instantiate a graph partition with a node for each solution. These nodes do not have edges between them. There are undirected edges between all other pair of nodes from different partitions. This generates a complete R-PARTITE graph with $m + n$ as R. The generated graph is shown in figure 8. Each edge has a cost which signifies the probability that the corresponding solutions are co-optimal. This



Complete R Partite Graph

Figure 8: Complete R-Partite Max Cost Clique

can be computed using equation 10. The problem is to find a clique in this graph with maximum cost. Note that the largest clique in this graph can be trivially generated by picking a node from each levels. The challenge is to generate the clique with maximum cost. Picking one node from each partition would ensure one solution for each starting T value is chosen. Hence the total number of chosen solutions are $R = m + n$. Larger the cost of the clique, higher the probability of co-optimality of all solutions. Hence a larger solution space could be represented by the same number of solutions.

Theorem: COMPLETE R-PARTITE MAX COST CLIQUE is NP-Complete
Proof: Transformation from 3SAT [9]. Rest of the proof is omitted for brevity.

Heuristic for Criteria 3

Although the approximate algorithm [14] is interesting, we found a heuristic (refer Algorithm 2) to perform better in our problem. We are given $m + n$ sets of solutions sorted in increasing order of T. We traverse these sorted set one by one and at each set we generate the possible cliques for each potential node in this current set by merging it with each of the best cliques generated at the previous set. From these potential solutions, the clique that gives the maximum cost is stored. When the next set is traversed, the possible solutions at the next set will use this best clique information. The is done iteratively till the last set is encountered. At this stage the clique with best cost is selected.

Algorithm 2 *Heuristic for Criteria 3*

INPUT: set of mn solution partitioned in $m + n$ distinct sets
Get the probability of co-existence between each solution pair i and j
Sort the $m + n$ sets depending on their starting T values
Loop(over all $m + n$ set)
  For(each solution x in set i)
    For(each solution y in set $i - 1$)
      Calculate the cost of the clique formed by adding x to the best clique at y
    Choose the clique with maximum cost and store it. This is the cost of the best clique at x now
At the last set, among all x in this set, pick the clique with largest cost

---

This completes the description of the three pruning criteria. Each of them has its own distinct property. **Criteria 1** ensures that the best solution will never be pruned out, but is not polynomial. **Criteria 2** has a methodology which enables more pruning and hence is faster than **Criteria 1** but is still not provably polynomial. **Criteria 3** is strictly polynomial and has a very firm selection mechanism.

Bench # Sinks	Van-Gin Sol (nano sec)	T_{cons} (nano sec)	Van-Gin P_e	# Bufs	Crit 1 P_e	# Bufs	Crit 2 P_e	# Bufs	Crit 3 P_e	# Bufs
54	121.2	122	0.198	14	0	13	0	15	0	13
96	102.4	103	0.563	19	0	19	0	19	0	20
216	585.9	586	0.467	23	-	-	0	23	0	20
360	117.7	118	0.917	70	-	-	0	73	0	72
468	582.6	583	0.531	45	-	-	0.025	41	0.213	32
590	390.8	391	0.618	77	-	-	0	80	0	79
720	722.3	723	0.596	61	-	-	0	63	0	59
890	845.5	846	0.573	60	-	-	0	59	0	57
1080	1598.3	1599	0.414	78	-	-	0	84	0	69
1260	5812.2	5813	0.639	95	-	-	0	93	0	94
Avg			0.552	-			0.003		0.021	

Table 1: Results from Experiments

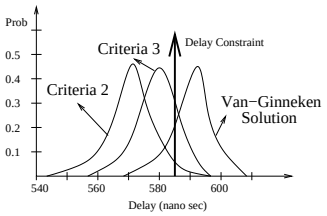


Figure 9: Comparison of Solutions for a Benchmark

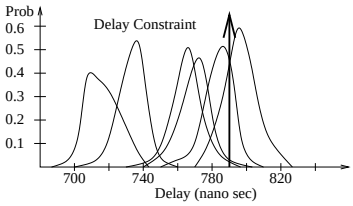


Figure 10: Delay Distribution of Solutions Satisfying a Delay Constraint

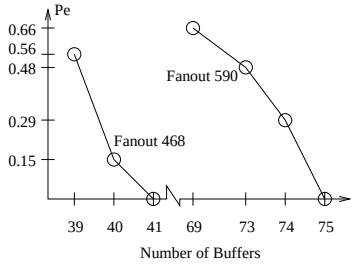


Figure 11: Trade-off Between Number of Buffers and Probability of Error

5. EXPERIMENTAL RESULTS

The objective through experimental results was to illustrate the superiority of our approach over fixed wire-length assumption and also compare the quality of the three pruning criteria. For experimental purposes, we used large wiring trees with large number of sinks. Some of these trees were balanced and some were unbalanced (close to lt-trees). The required arrival times at the sinks of the benchmarks were also chosen randomly. Values for r and c (wire parasitics) were chosen for 0.18 micron technology. Wire-lengths were taken as Gaussian distributions with mean varying between 100 to 1000 λ and variance lying within 10% of the mean. The Van-Ginneken algorithm was used to generate a valid buffer placement with largest required time at source. The delay constraint for the tree was set at a value slightly greater than the single delay value given by the Van-Ginneken Algorithm. This ensures that the Van-Ginneken estimate always satisfies the constraint.

Table 1 illustrates the performance of our criteria as compared with fixed wire-length estimate buffer insertion [13]. The average wire-length of the distribution was provided as wire length estimate input to the Van-Ginneken algorithm. We chose a solution from the Van-Ginneken algorithm that satisfied the delay constraint and then generated the delay distribution for the solution. The second column of table 1 provides the output of the Van-Ginneken algorithm. It can clearly be seen from the results that fixed value estimates result in delay constraint violation with very high probabilities (P_e). Given a delay constraint P_e is computed using $P_e = \sum_{d \geq D_{cons}} p(d)$

On the other hand probabilistic buffer insertion results in the delay constraint being satisfied with a very high probability. Even though the total number of buffers between our approach and the Van-Ginneken approach remain more or less the same, it was observed that the buffer locations are different. Another observation that we made was that Criteria 1, which has a very relaxed pruning strategy was not practical for larger benchmarks since it did not give results in a reasonable run time. Results also show that Criteria 2 solutions have the least probability (P_e) of not satisfying the delay constraint without any major buffer penalty. We also observed that for most cases, the actual delay distribution found using our criteria were much better than those from the Van-Ginneken fixed value algorithm. This is illustrated in figure 9. Clearly, the best solution of the Van-Ginneken approach is worse than that generated by Criteria 2. There is an important inference here: **optimization using fixed average wire-length estimates does not generate a distribution with the smallest average delay**. This is a critical observation since optimization using average wire-length values does not optimize average delay at the root.

Figure 10 shows a typical delay distribution solution at the root

using the probabilistic approach. It is evident that there are several possible solutions for a given delay constraint (shown by the arrow in figure 10) that have differing probabilities of satisfying the constraint. We observed from the experiments that a trade-off exists between probabilistically satisfying a delay constraint and the number of buffers used by that solution. Figure 11 shows such a trade-off for two benchmarks. This gives the designer the flexibility of taking a $Probabilistic - Risk$ of not satisfying the delay constraint while choosing a solution with fewer buffers.

Table 2 shows that the run time comparison between the three Criteria proposed in section 4. All values have been normalized w.r.t. the run time values for Criteria 2. Criteria 1 is not polynomial in complexity and hence has a very large run time for reasonably large benchmarks. It has a much weaker pruning criteria, but the quality of final solution could be very high since it retains all possible solutions which can potentially be better. But the runtime for Criteria 1 was unreasonably large making it impractical when compared with Criteria 2 and 3. As can be seen from table 2, run time for Criteria 2 is similar to that of Criteria 3 even though it is not polynomial in the worst case. This illustrates that Criteria 2 has a very good performance both in terms of run time and quality of the final solution.

Bench Sinks	Criteria 1	Criteria 2	Criteria 3
54	2.39	1	1.02
96	3.75	1	1.03
216	-	1	0.90
360	-	1	1.57
468	-	1	1.05
590	-	1	0.97
720	-	1	0.92
890	-	1	0.95
1080	-	1	0.91
1260	-	1	1.12

Table 2: Runtime Comparison Between the Three Criteria

6. CONCLUSION AND FUTURE WORK

Estimation in design automation is marred with inaccuracies. This paper address the problem of wire-length estimation unpredictabilities in buffer insertion. It solves the problem by modeling the wire-lengths as probability distributions and proposes algorithms for solving them. Traditional approach for buffer insertion is compared with this novel approach. Results showed that traditional solution can result in a very large probability of error while our algorithms resulted in very accurate solutions. Several results of runtime complexity were also discussed.

The future work would include developing models for estimating wire-length distributions and developing a complete synthesis system that probabilistically traverses the solution space.

7. REFERENCES

[1] A. Agarwal et al. "Computation and Refinement of Statistical Bounds on Circuit Delay". In *Proc. Design Automation Conference*, June 2003.

[2] A. Davoodi and A. Srivastava. "Voltage Scheduling Under Unpredictabilities: A Risk Management Paradigm". In *To appear in ACM/IEEE Int'l Symposium on Low Power Electronics and Design*, August 2003.

[3] A. Srivastava and Majid Sarrafzadeh. "Exact Algorithm for Modifying Buffer Trees Using Buffer Duplication in a Delay Optimization Perspective". In *International Workshop on Logic Synthesis*, 2001.

[4] A. Srivastava and Majid Sarrafzadeh. "Predictability: Definition, Analysis and Optimization". In *Procs of International Conference on Computer Aided Design*, Nov. 2002.

[5] C. Visweswariah. "Death, Taxes and Failing Chips". In *Proc. of Design Automation Conference*, June 2003.

[6] C.L. Berman, J.L. Carter and K.F. Day. "The Fanout Problem: From Theory to Practice". In *C.L. Seitz, editor, Advanced Research in VLSI: Proceedings of the 1989 Decennial Caltech Conference*, pages 69–99. MIT Press, March 1989.

[7] D. Stroehle. "Avoiding The Pitfalls in CMOS Design". In *New Electronics*, page 30, June 1987.

[8] G. Tellez and Majid Sarrafzadeh. "Clock Period Constrained Minimal Buffer Insertion in Clock Trees". In *Proceedings of the International Conference on Computer-Aided Design*, 1994.

[9] M.R. Garey and D.S. Johnson. *Computers and Intractability, A guide to the theory of NP Completeness*. W.H. Freeman and Company, New York, 1979.

[10] J. Hu, Sachin Sapatnekar. "Simultaneous Buffer Insertion and Non-Hanan Optimization for VLSI Interconnect under a Higher Order AWE Model". In *Proceedings of the ACM International Symposium on Physical Design*, 1999.

[11] J.D. Cho and Majid Sarrafzadeh. "A Buffer Redistribution Algorithm for High-Speed Clock Routing". In *Proceedings of 1993 IEEE Design Automation Conference*, 1993.

[12] A. Srivastava E. Kursun and M. Sarrafzadeh. "Predictability Driven Binding: Methodologies and Tradeoffs". In *Journal of Circuits, Systems and Computers, Special Issue on Low Power IC Designs*, 2002.

[13] L.P.P.P. van Ginneken. "Buffer Placement in Distributed RC-tree Networks for Minimal Elmore Delay". In *Proc of International Symposium on Circuits and Systems*, pages 865–868, December 1990.

[14] S. Arora. "The Approximability of NP-Hard Problems". In *ACM STOC*, 1998.

[15] S. Borkar et al. "Parameter Variations and Impact on Circuits and Microarchitecture". In *Proc. Design Automation Conference*, June 2003.

[16] W.C. Elmore. "The Transient Analysis of Damped Linear Networks with Particular Regard to Wideband Amplifiers". In *Journal of Applied Physics, vol. 19(1)*, 1948.

[17] Yanbin Jiang, Sachin Sapatnekar et al. "Combined Transistor Sizing with Buffer Insertion for Timing Optimization". In *Proceedings of the IEEE Custom Integrated Circuits Conference*, 1998.