Minimal Logic Re-synthesis

Gitanjali M. Swamy Sriram Rajamani Chris Lennard

Robert K. Brayton

Abstract

Most problems in logic synthesis are computationally hard, and are solved using heuristics. This often makes algorithms un-stable; if the input is changed slightly, the new result of synthesis can be significantly different. A designer can spend much effort hand-optimizing a circuit, so it is desirable to retain as much of this human insight as possible. This motivates the need for incremental synthesis. We propose a re-synthesis algorithm, which allows the designer to designate non-resynthesizable portions of a circuit. We define the concept of minimal change caused by re synthesis, i.e. given a functional change to the circuit, we examine the minimal change to implement this change. For the evaluation of a region for re-synthesis we present techniques for evaluating the "sensitivity" or gain possible with re-synthesis of a set of nodes. We conclude with experimental results and future directions.

1 Introduction

Logic synthesis refers to the process of optimizing a logic description of a circuit, given as a net-list of logic (boolean) gates [1]. This representation can be optimized for area(minimum), delay (minimum or meeting requirements), and power(minimum). Since these problems are hard to solve exactly, heuristic algorithms are generally used. However, these algorithms are unstable; if a small change is made in the network function, the output of the synthesis algorithm may vary greatly from the previous implementation. A designer can invest effort in optimizing the original design by hand, so it is desirable that the hand-designed or optimal parts be preserved, even when changes are made to the specification. In addition, the network may have already been implemented in silicon at a lower level of the design hierarchy, and it can be inconvenient to change.

Previous algorithms for the problem of incremental synthesis have dealt with post-rectification (Watanabe et al. [2]), and preserving cones of logic (Brand et al.[3]) in the design. Some relevant work has also been done by Kukimoto and Fujita [4] but this is concerned with FPGA's rather than general logic. In addition, this work restricted re-synthesisizable parts of the network to all nodes at a level, rather than general re-synthesis region. Other approaches to this problem, which use boolean unification were proposed by Fujita et al [5], and Lin et al [6], however these approaches do not consider the optimality of sub-regions in the network as a factor in choosing candidate regions for re-synthesis. None of the above approaches have dealt directly with preserving the "highly-optimal" parts of the circuit.

Changes to the system are defined as changes in the functions computed at the primary output nodes. The problem is stated as: We are given a logic design that has inputs $x_i \in x$, outputs $z_i \in z$, and an implementation I of $z = F(x)$. I has already been optimized for an objective, which may be power, delay or area. A new specification is given, $z = F_{\text{new}}(x)$. We assume that F and F_{new} are completely specified; the extensions to incomplete specifications are straightforward. The designer can designate which regions may not be resynthesized, or order the regions in terms of where re synthesis is more acceptable.

We use an evaluation criteria for the acceptability of regions for re-synthesis called sensitivity is generated. In this paper we compute the sensitivity (or acceptability for re-synthesis) for power. In this respect, we rely heavily on the work done by Lennard [7] for the computation of power sensitivities of nodes.

We propose an iterative solution to the problem: we begin with a small region for re-synthesis (selected using the sensitivity criteria), and iteratively expand that region until a solution is obtained. At each stage, we test if this re synthesis region alone can realize the new specification. As a second pass, we trim the region iteratively, so that it becomes minimal in the sense that no subset of the current region can alone realize the change in functionality.

This paper is organized as follows: Section 2 describes the terminology and definitions in this paper. In Section 3 we give a procedure for determining whether re-synthesis of a given a sub-region of the network can realize an implementation with the new functionality $z = F_{\text{new}}(x)$. In Section 4, we examine strategies for determining nodes in the network that are good candidates for the re-synthesis region. A good candidate for re-synthesis must give gains in the designers objective (power, area, delay) as well as implement the new functionality. This goodness is estimated by its sensitivity. Section 5 gives an iterative algorithm for incremental synthesis that begins with an empty re-synthesis region, and iteratively picks nodes from the rest of the network to add to the region (in order of their sensitivity). We present some experimental results in Section 6, and conclude with directions for future work in Section 7.

2 Terminology and Notation

- A **completely specified** Boolean function F with n inputs and m outputs is a mapping $F : B^n \longrightarrow B^m$, where $B = \{0, 1\}$. If $m = 1$ the **onset** and **offset** are the set of points satisfying $F(x) = 1$ and $F(x) = 0$ respectively. A **minterm** v of a function F is a vertex (i.e. a point in B^n) such that $F(v) = 1$. The **cofactor** F_{x_i} of F (completely specified) with respect to variable x_i is the function F evaluated at $x_i = 1$. The **Shannon** form of

$F = x_i \cdot F_{x_i} + \overline{x_i} \cdot F_{\overline{x_i}}$ (i.e. in terms of its cofactors). The **size** of $F(x)$, ($|F(x)|$) denotes the number of minterms (onset points) in $F(x)$. An **incompletely specified** Boolean function is a mapping $F : B^n \longrightarrow Y^m$ where $Y = \{0, 1, \star\}$ ($\star \Rightarrow F$ can be 0 or 1). If $m = 1$ the **onset**, **offset**, and **don't care set (dcset)** are the set of points such that $F(x) = 1$, $F(x) = 0$, and $F(x) = \star$ respectively.

- A **Boolean network** (Figure 1) $\mathcal{N}$, is a directed acyclic graph (DAG) such that each node in $\mathcal{N}$ has a Boolean function ($n = f_n(n_1 \ldots n_m)$). There is a directed edge from node n_i to node n if the function f_n is dependent on node n_i; node n_i is a **fanin** of a node n, node n is a **fanout** of node n_i. A node n_i is a **transitive fanin** of a node n if there is a directed path from n_i to n; n is a **transitive fanout** of n_i. The inputs $\mathbf{x} = (x_1, \ldots, x_n)$ of the Boolean network are called **primary inputs** and outputs $\mathbf{z} = (z_1, \ldots, z_m)$ are called **primary outputs**. Nodes with at least one fanin and one fanout are called **internal**. The projection ($\Omega_I(S(n))$) is the representation of set $S(n)$ in terms of the variables of some input set I (in general I denotes the inputs the node in concern, i.e if $n = G(I)$, then $\Omega_I(S(n)) = G^{-1}(S(n))$).

- An **Observability Relation** is a mapping $O^F(x, z) : B^n \times B^m \longrightarrow B$, where x are inputs and y outputs. Given any function of the network with inputs $x = (x_1, \ldots, x_n)$ and outputs $z = (z_1, \ldots, z_m)$, ($z = F(x)$) may also be represented as its **observability relation** $O^F(x, z) := z \overline{\oplus} F(x)$. Given inputs x and outputs z, an observability relation is characterized as $O^F(x, z) = 1$ if $z = F(x)$ and $O^F(x, z) = 0$ if $z \neq F(x)$.

- **Consensus or universal quantification** $\forall$ is defined as $\forall_{x_i} f(x_1, \ldots x_n) = f_{x_i} \cdot f_{\overline{x_i}}$. It is the largest boolean function contained in f that is independent of x_i.

  Smoothing or existential quantification $\exists$ is defined as $\exists_{x_i} f(x_1, \ldots, x_n) = f_{x_i} + f_{\overline{x_i}}$. It

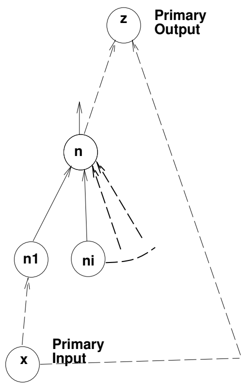

Figure 1: Network

is the smallest Boolean function containing f that is independent of x_i.

- The **Boolean difference** of a function f with respect to a variable x is defined as $\frac{\partial f}{\partial x} \equiv f_x \overline{f}_{\overline{x}} + \overline{f}_x f_{\overline{x}}$. This function gives all the conditions under which the value of f is **influenced** by the value of x. Its complement therefore is all the conditions under which f is **insensitive** to x.

3 Conditions on a Valid Re-synthesis Region

As stated before, we have a logic function (possibly multi-output) $z = F(x)$ ($z = (z_1 \ldots z_m)$), which is a function of inputs $x = (x_1 \ldots x_n)$. We already have a network realization for this function. The logic function of this network is changed to a new function $z = F_{\text{new}}(x)$. We may also represent the new function as a separate network. The objective is to realize $F_{\text{new}}(x)$ while simultaneously preserving as much of the old network structure I (particularly hand-optimized portions) as possible.

We recognize the following sub-problem: Given a network with original functionality $z = F(x)$, which is to be changed to $z = F_{\text{new}}(x)$, and a region for re-synthesis R (see figure 2) with inputs v and outputs u, determine whether the new function can be implemented by re-synthesizing the region R exclusively.

To answer this question, we first compute an observability relation for the region R, that is consistent with the overall implementation $z = F_{\text{new}}(x)$ and compatible with the implementations for the remainder of the original network. Next we impose conditions on this relation that ensure that it is implementable.

3.1 Computing the Observability Relations

The overall observability relation for the original circuit is characterized by $O^F(x, z) = z\overline{\oplus}F(x)$, and the new relation is characterized by $O^F_{\text{new}}(x, z) = z\overline{\oplus}F_{\text{new}}(x)$. The region of re synthesis R, with inputs v and outputs u, is characterized in its current implementation, by an observability relation $O^R(v, u)$ that is consistent with $O^F(x, z)$ and compatible with the remainder of the network. The remainder of the network is characterized by the network N (with the region R deleted), with inputs u and x and outputs z and v. Its characteristic function is given by $N(x, v, u, z)$. Figure 2 illustrates these regions, and their inputs and outputs. In this section we illustrate how to compute the required functionality of a predefined region when the output function is changed. We first state the following theorem, which is adapted from [8].

Theorem 3.1 *The observability relation for region R that is consistent with $N(x, v, u, z)$ and compatible with $O^F(x, z)$ is:*
$O^R(v, u) = \forall_{x,z}(N(x, v, u, z) \Rightarrow O^F(x, z))$.

When F has been changed to F_{new}, we can simply replace F by F_{new} in the above and give a condition for realizability of the new functionality.

Theorem 3.2 *Let $O^R{}_{new}(v, u) = \forall_{x,z}(N(x, v, u, z) \Rightarrow O^F_{new}(x, z))$. The new functionality can be realized by re-synthesizing R iff $(\forall_v \exists_u O^R{}_{new}(v, u) = 1)$,*

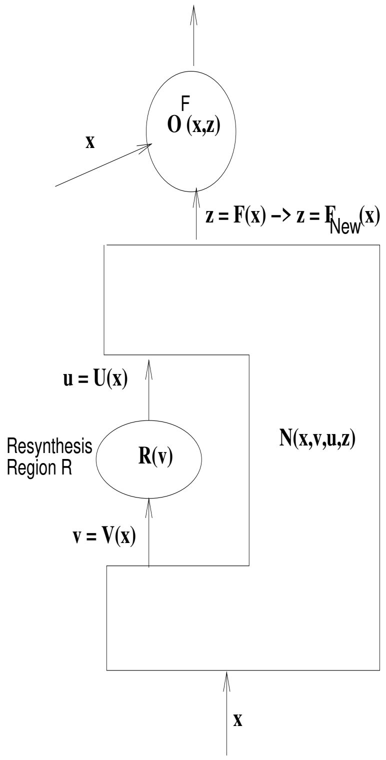

Figure 2: Network

A relation that satisfies Theorem 3.2 cannot directly yield a hardware realization, since hardware can only implement functions. In general, $O^R{}_{\text{new}}(v, u)$ is not a function. However, any $O^R{}_{\text{new}}(v, u)$ satisfying the theorem has at least one function as a subset. To find such a function, we have to solve a set of boolean equations. The following theorem from boolean unification [9], details all solutions $y = G(x)$ to the equation $f(x, y) = 1$, when y is a single bdd variable.

Theorem 3.3 *The solutions $y = G(x)$ to an equation $f(x, y) = 1$, where y is a single variable, can be characterized by the inequalities $\overline{f(x,0)} \subseteq G(x) \subseteq f(x, 1)$.*

The following theorem, adapted from [9], characterizes a family of functions that yield a valid implementation of the relation. We assume that $O^R{}_{\text{new}}(v, u)$ satisfies Theorem 3.2. Note that Theorem 3.4 is a generalization of Theorem 3.3 for multiple output variables.

Theorem 3.4 *If $O^R{}_{new}(v,u)$, $u = u_1 \ldots u_m$ is the observability relation for a region R, then any function $u = R(v) = g_1 \ldots g_m$ that satisfies:*

- $f_i + d_i = FOC_i = (\exists_{\ldots u_j \cdots_{(j \neq i)}} O^R{}_{new}(v,u)) \cdot \prod_{k<i}(u_k \overline{\oplus} g_k(v))|_{(u_i=1)}$

  $r_i + d_i = FRC_i = (\exists_{\ldots u_j \cdots_{(j \neq i)}} O^R{}_{new}(v,u)) \cdot \prod_{k<i}(u_k \overline{\oplus} g_k(v))|_{(u_i=0)}$

  $f_i \leq g_i \leq f_i + d_i$

satisfies the relation $O^R{}_{new}(v,u)$. (i.e, $R(v) = u \Rightarrow O^R(v,u) = 1$). At every stage f_i represents the onset, d_i the don't care set, and r_i offset from which g_i is chosen.

We can use the freedom provided by the d_i to optimize the function. Reordering the u_i gives rise to different functions; in general a good ordering should be found.

4 Sensitivity

The sensitivity of a node (or region) in the circuit is defined according to the choice of objective (area, delay, power).

Definition 1 *The sensitivity of a node is the change in the objective function (area, delay or power) that is expected if the node is re-synthesized.*

The definition of sensitivity is easily extended to regions. We reply on the work done by Lennard [7] et al. on computing the power sensitivity of a node. A node n is a good candidate for re-synthesis if local change in activity (power) plus change in activity in the transitive fanout reduces overall power. A method for determining expected activity $E(n)$ is outlined in [7].

Consider a node n in the network with immediate fanins $n_1 \ldots n_m$ (refer to Figure 1). Node n computes a function $f_n(n_1 \ldots n_m)$ of its fanins. Let A_{n_1} denote an arbitrary set of minterms that are added to the onset of fanin n_1. Let this be the only change made to the fanins of n. The set of of minterms that are added to the onset of f_n are those minterms in A_{n_1} that actually change the value of the function f_n from 0 to 1. Similarly, the set of minterms that are removed from the onset of f_n are those minterms in A_{n_1} that actually change the value of f_n from 1 to 0. Let A_n denote the set of minterms that are added to the onset of f_n and R_n denote the set of minterms that are removed from the onset of f_n due to this change. A_n and R_n can be computed as follows:

1. $A_n(A_{n_1}) = \mathbf{\Omega}_I(f_n|_{n1=1} \cdot \overline{f_n|_{n1=0}}) \cdot A_{n_1}$.
2. $R_n(A_{n_1}) = \mathbf{\Omega}_I(\overline{f_n|_{n1=1}} \cdot f_n|_{n1=0}) \cdot A_{n_1}$.

The quantities $S^p{}_n(n_1) = f_n|_{n1=1} \cdot \overline{f_n|_{n1=0}}$, and $S^n{}_n(n_1) = \overline{f_n|_{n1=1}} \cdot f_n|_{n1=0}$ are called the functional positive and negative sensitivities. Similar measures can be computed for set of minterms that are added and subtracted from the onset of f_n, when R_{n_1} minterms are subtracted from the onset of its fanin.

These quantities have no real significance as yet, since we do not know the exact change that is actually made to an internal node of the network. However, if we assume that *any change in the onset size is equally likely*, the expected size of the sets A_n and R_n can be computed with just the knowledge of the size of the change (without knowing the actual minterms in the change!) by computing the expectations of the quantities in equations (1) and (2):

- $E(|A_n(A_{n_1})|) = |A_{n_1}| \frac{|\Omega_I(S^p{}_n(n_1)) \cdot \overline{f_{n_1}}|}{|\Omega_I(\overline{f_{n_1}})|}$
- $E(|R_n(A_{n_1})|) = |A_{n_1}| \frac{|\Omega_I(S^n{}_n(n_1)) \cdot f_{n_1}|}{|\Omega_I(f_{n_1})|}$

Given a probability p of evaluating to high at a node, the functional transition activity is given by 2(1-p)p. Thus, a p far from 0.5 implies a smaller transition activity. Given a change in onset size at a given node n_1 , the the expected change in onset size can be derived for all nodes in the transitive fanout of n_1. The expected onset sizes of nodes directly relate to their switching probabilities and hence can be used to get a measure of the expected change in power for the circuit.

We use this analysis to compute a "good" ordering of nodes for re-synthesis. The efficacy of this measure of the sensitivity of a node has been demonstrated statistically in [7]. For the purpose of this paper, we will not discuss this further.

5 Iterative Algorithm

We combine the method for determining whether a region is sufficient to realize the new functionality (Section 3), and the means of evaluating which nodes to add to the re-synthesis region (Section 4), and propose an iterative algorithm. Before we give the details of the iterative algorithm, we impose certain restrictions on the structure of valid regions.

We require a loose form of structural contiguity restriction on R. In particular, we do not allow the inputs of R to be dependent on outputs from R. This structural restriction is needed to use Theorem 3.2. For instance in Figure 3, R, which is composed of two non-contiguous regions,

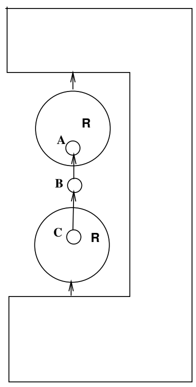

Figure 3: Invalid Regions

is not a valid region, since input A depends upon output C. This restriction is imposed, since resynthesis of such a region might possibly create a combinational cycle within the network.

5.1 Searching for Minimal Regions

Definition 2 *A resynthesis region is minimal if no node can be removed from R without destroying* *the ability, by re-synthesis of R, to obtain the new functionality F_{new}.*

It is easy to see that, if a region R is sufficient for re-synthesis (Theorem 3.2), then every superset $R \subseteq R'$ is also sufficient for re-synthesis. Consequently, if we greedily remove nodes from a feasible region, while maintaining feasibility and structural validity, our search is guaranteed to terminate in a locally minimal region. However, finding a global minimum is much harder. We are currently examining a strategy that implicitly enumerates all feasible sub-regions and is guaranteed to find a global minimum.

5.2 Iterative Algorithm

First, the designer is allowed to mark out regions that may not be re-synthesized. For the remaining network, we compute the approximate sensitivities for every node in the network. Computing the sensitivity of a node requires determining which of its transitive fanins may be re-synthesized. First a quick measure of sensitivity is computed to estimate the region that must be re-synthesized with the node. We use an iterative algorithm that progressively adds nodes to the re-synthesis region.

Algorithm 5.2

Mark regions designer wants unchanged (M)

$R = \phi$

While R not sufficient (Theorem 3.2)

  p = Node of highest sensitivity (excluding M and R)

  $R = R + p$

Reduce R to get a *minimal* region (Section 5.1)

Compute $u = R(v)$ (Theorem 3.4)

Synthesize $u = R(v)$ to get R_{new}

replace R with R_{new}

return

We add the node of highest sensitivity to the resynthesis region R, and examine whether re-synthesizing the current region R could achieve the new functionality. If not, we add the nodes of next highest sensitivity to the region and repeat the process. This greedy process iteratively tries to determine a small partition of the initial circuit that has high potential for gains in the objective (power for our current implementation) that can be re-synthesized in order to implement the new functionality. In order to make the region minimal (Section 5.1, we post process it by attempting to remove nodes to get a smaller feasible region.

6 Experiments and Results

We have implemented the iterative algorithm and sensitivity measures described in this paper in SIS [1]. Though not explicitly stated during this paper, we used the BDDs to represent our functions and relations. Logical predicates may be represented as a sequence of BDD operations; we used BDD's for the computation of logical predicates.

We had to design a set of experiments where we emulated the design process on which these methods would be applied. We assume that an original design has been hand optimized for the design objective (in this case power) and that a new specification has been given. The designer, feeling that the old design is not only highly optimal, but maybe has been tuned to some other objectives or has already been physically designed, wants to keep as much of this around as possible.

To emulate a design change, we took benchmark examples with external don't cares and optimized them to obtain a circuit called "old". The new spec F_{new} for each example is the same circuit benchmark but without the external don't cares given. If we optimize this without trying to preserve any of "old", we get a circuit "new". This represents the best we could do with the new specification, but most of "old" would have been changed. Note that "new" should always be worse than "old" in this scenario, in terms of power.

Our primary objective is to preserve as much of the old implementation as possible by re-synthesizing the re-synthesis region alone. However, we would also like to get “good” power results while preserving as much of the old network structure as possible.

In general our primary goal conflicts with an objective of minimal power; we can always get better power results by completely ignoring "old" (hence we have more flexibility). Thus we expect to get a power number for the incremental approach which is worse that for "new". However, using a good measure of power sensitivity to pick nodes to add to the re-synthesis region should give us better power results than using any random method to pick the nodes. There may be many minimal regions in a network that can implement the same functional change; our objective is to pick the minimal region that gives good power results.

In our experiments, we used two methods for choosing the region for re- synthesis: "sensitivity" and "random". In "sensitivity", we chose the new node to be put in the region of re synthesis according to the power sensitivity measure discussed in Section 4. In "random" we chose the new node randomly. Note that both approaches do not take the ability of a node to resynthesize the new function when choosing a node. The iterative algorithm (Section 5.2) and the greedy search for a minimal region (Section 5.1) attempt to minimize the resynthesis region. We compared the sizes of the regions, as well as total power of the resultant network obtained for both these measures.

Figure 4 summarizes the percentage of the network preserved; these numbers were consistently over 50% and quite often as high as 90%. We were indeed preserving large portions of the old network. However, at this stage we do not know what the exact minimum answer is; our future work will determine this.

We also tabulate the results of our experiment on some sample examples in Table 1. Isyn denotes the results obtained by our incremental synthesis and Nsyn denotes results obtained by complete resynthesis. In 10 out of 14 examples,"sensitivity" outperformed "random". We see that for the first ten examples in Table 1, (alu3 through x1dn), "sensitivity" produces circuits with better power numbers than "random", mostly with smaller or equivalent sized regions. We were chosing re-synthesis regions that were both small as well as better in power (as compared to a random criteria). For the last four examples, "random" produces circuits with lower power, sometimes at the cost of larger re-synthesis regions. Note that in all but two examples (cmb and sao2), lower power numbers correspond to smaller regions of re-synthesis. This is surprising since re-synthesizing the entire network leads to lowest power. However, this can be attributable to the fact that script.rugged does not necessarily produce a circuit of minimal power. As part of future work, we plan to examine a better power synthesis routine. The power numbers were computed assuming a $20Mhz$ clock and a Vdd of $5v$.

7 Conclusions and Future Work

Given an original network and a changed specification, we have shown how to realize the new specification while preserving much of the old network. In particular, we have defined and used a measure of power sensitivity of the node, and shown that by choosing nodes for re-synthesis according to this , we get mostly better results than any random selection of nodes. Our method of re-synthesis is effective in preserving much of the old network. It is also effective in picking the

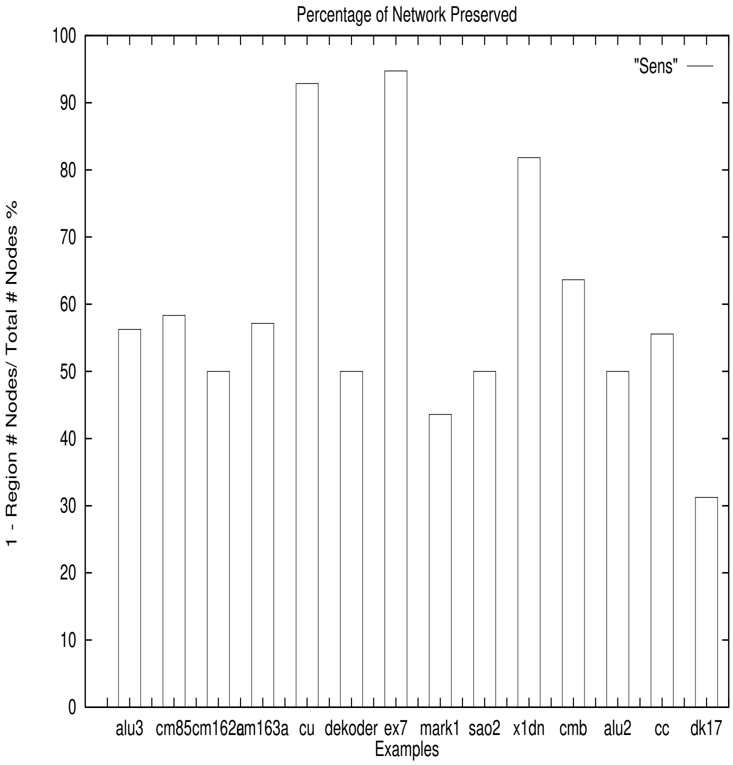

Figure 4: Size of the Re-synthesis Region

	♯ Total	♯ Region		ISyn Power		NSyn Power	
Example	Nodes	Sensitivity	Random	Sensitivity	Random	Old	New
alu3	16	7	8	643.10	716.40	389.00	435.10
cm85a	7	3	4	172.90	177.30	189.10	201.30
cm162a	12	5	7	202.00	273.70	163.40	177.30
cm163a	8	4	5	173.10	203.90	169.20	179.60
cu	14	1	1	214.60	224.60	214.60	214.60
dekoder	10	5	6	205.00	214.60	159.90	183.10
ex7	19	1	1	535.20	575.80	535.20	535.20
mark1	39	22	28	589.50	657.60	281.70	532.60
sao2	20	10	9	671.30	759.50	658.60	674.80
x1dn	11	2	8	483.70	640.50	435.90	443.80
cmb	8	4	5	249.20	234.20	229.30	256.50
alu2	22	8	6	1189.70	872.90	360.90	492.80
cc	18	8	5	263.80	242.80	212.50	222.40
dk17	16	11	11	360.10	353.60	258.00	274.70

Table 1: Results

re-synthesis region so as to get good power results (as compared to any other random strategy).

As part of future work, we plan to examine different measures of the sensitivity of a node (region) (wrt to different objectives), and evaluate the performance of the iterative algorithm using these strategies. In this paper, we have described a greedy strategy for re-synthesis, however in the future we plan to examine other alternate strategies. We are currently implementing an exact formulation for the minimal re-synthesis region to realize a given change. We want to come up with a related greedy strategy that uses some form of sensitivity wrt to a nodes ability to achieve re-sythesis.

In Section 3 we implemented one particular function from the entire class of possible functions. We intend to extend this and examine the entire class of solutions for the most optimal implementation, using the work of Watanabe et al [10] on heuristic boolean minimization. Since we are recomputing a new ${O^R}_{\text{new}}(v, u)$ many times during the iterative algorithm, it becomes pertinent to explore incremental ways of updating the relation, rather than re-computing it from the beginning. We expect that some of the methods adapted from [11] may be used.

References

[1] E. M. Sentovich, K. J. Singh, C. Moon, H. Savoj, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "Sequential Circuit Design Using Synthesis and Optimization," in *Proc. Intl. Conf. on Computer Design*, pp. 328–333, Oct. 1992.

[2] Y. Watanabe and R. K. Brayton, "Incremental Sythesis for Engineering change," in *Workshop Notes of the Intl. Workshop on Logic Synthesis*, (Tahoe City, CA), May 1991.

[3] D. Brand, "Incremental Synthesis," in *Proc. Intl. Conf. on Computer-Aided Design*, pp. 126–129, Nov. 1992.

[4] Y. Kukimoto and M. Fujita, "Rectification method for lookup-table type FPGA's," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pp. 54–61, November 1992.

[5] M. Fujita, Y. Tamiya, Y. Kukimoto, and K.-C. Chen, "Application of Boolean unification to combinational logic synthesis," in *Proceedings of IEEE International Conference on Computer-Aided Design*, pp. 510–513, November 1991.

[6] C. Lin, K. Chen, S. Chang, M. Marek-Sadowska, and K. Cheng, "Logic Synthesis for Engineering Change," in *Proc. of the Design Automation Conf.*, pp. 647–652, June 1995.

[7] C. Lennard, *Estimation Techniques to Guide Low Power Resynthesis Algorithms For Combinational Random CMOS Logic*. PhD thesis, University of California Berkeley, Electronics Research Laboratory, College of Engineering, University of California, Berkeley, CA 94720, Aug. 1995. Memorandum No. UCB/ERL M95/75.

[8] H. Savoj and R. K. Brayton, "Observability Relations and Observability Don't Cares," in *Proc. Intl. Conf. on Computer-Aided Design*, pp. 518–521, Nov. 1991.

[9] F. M. Brown, *Boolean reasoning : the logic of Boolean equations*. Boston : Kluwer Academic Publishers, 1990.

[10] Y. Watanabe and R. K. Brayton, "Heuristic Minimization of Multiple-Valued Relations," *IEEE Transactions on Computer-Aided Design*, vol. Vol. 12, pp. 1458 – 1472, October 1993.

[11] H. Savoj, *Don't Cares in Multi-Level Network Optimization*. PhD thesis, University of California Berkeley, Electronics Research Laboratory, College of Engineering, University of California, Berkeley, CA 94720, May 1992.