Multi-way Partitioning For Minimum Delay For Look-Up Table Based FPGAs

Prashant Sawkar and Donald Thomas

Electrical and Computer Engineering Dept.
Carnegie-Mellon University
Pittsburgh, PA 15213

Abstract

*In this paper we present a set cover based approach (***SCP***) to multi-way partitioning for minimum delay for Look-Up Table based FPGAs. SCP minimizes the number of chip-crossings on each circuit path with minimum logic duplication costs to achieve implementations with minimum delay and minimum number of chips. The overall complexity of* **SCP** *is* $\bigcirc(V^2)$. *Experimental results demonstrate that* **SCP** *produces partitions that on the average have 14% fewer chips, 28% fewer pins, and 93% fewer chip-crossings on each circuit path compared to* **ANN** *which is a simulated annealing based implementation of classical multi-way partitioning.* **SCP** *achieves this performance and compact packing at the cost of duplicating 13% of logic on the average. Additionally, in comparison with a lower bound we observe that* **SCP** *has produced near-optimal solutions.*

1 Introduction

Partitioning plays an important role in all aspects of VLSI design, ranging from logic synthesis, layout, testing, simulation and verification, and packaging at the device, board, and system levels. Several approaches have been proposed on partitioning for FPGAs [1, 2, 3]. Kuznar [1, 2] associates a $cost to each FPGA device and uses a bi-partitioner recursively to arrive at a multi-way partition that minimizes the $cost and interconnection costs of the partition. Sechen [3] first identifies clusters, and then uses a simulated annealing based placement of clusters on to an MCM to minimize wirelength and timing penalty.

32nd ACM/IEEE Design Automation Conference ®

However, the number of paths examined during annealing for timing violations is limited due to run-time considerations.

The important factors that govern performance of a single chip design are the depth of logic and wire-delays. Additionally, when a design is to be partitioned, the number of chip crossings (also referred to as path cuts elsewhere) on a circuit path becomes a dominating source of delay due to significantly higher delays that are experienced due to I/O buffers and higher capacitances encountered on external wiring. In this paper we present an approach to partitioning that minimizes the number of chip crossings on each circuit path.

2 Problem Formulation

A combinational boolean network **N** can be represented as a directed acyclic graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$; the vertices in **V** represent circuit elements (or LUTs), and edges in **E** represent signal nets. A primary input (PI) node has no incoming edges and a primary output (PO) node has no outgoing edges. Our assumption of a combinational logic input to partitioning is not a limitation. When we are given a general boolean network, the sequential elements are ignored temporarily and are re-assigned to the appropriate chips after partitioning. We define the **ISET** to be composed of the PIs of the original network, and also the outputs of sequential logic. Similarly, we define the **OSET** to be composed of the POs of the original network augumented by the inputs of sequential logic.

In this paper we present a new constructive approach to multi-way partitioning for minimum delay on to FPGAs. Given an input network $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ and an FPGA device $\mathcal{D}$, we construct a multi-way partition which has the fewest number of chip crossings on any circuit path and the fewest number of chips.

3 Partitioning For Minimum Delay

The primary goal of our partitioning formulation is to minimize the number of chip crossings on each circuit path. The secondary goal is to minimize the number of chips. Towards this end we propose a three phased approach:

1. Generate a family of clusters $\mathcal{F}$ that attempt to minimize chip crossings.
2. Generate a Cover $\mathcal{C} \subseteq \mathcal{F}$ which covers the starting network $\mathbf{G} = (\mathbf{V}, \mathbf{E})$.
3. Pack the clusters in the cover $\mathcal{C}$ into minimum number of chips.

4 Clustering Phase

Clustering plays an important role in partitioning by contracting the original netlist by way of identifying strongly connected components. The classical approaches to clustering discussed in [5] minimize total number of nets cut or number of clusters on a circuit path and are not suited well for minimizing the number of chip crossings on individual circuit paths. In order to minimize the number of chip crossings on every circuit path we have devised a new approach based on clustering one or more cones of logic, where each cone may have an element of the **OSET** as the root and consist of nodes in the root's transitive fanin. This type of clustering has the advantage of loacalizing all the paths from the cluster's inputs to the cluster's outputs. An important structural component in the input network is a multi-fanout net. Corresponding to each multi-fanout net there exists a cluster corresponding to the cones of the terminal nodes ($node \in \mathbf{OSET}$) encountered in its fanout cone.

The goal of the clustering phase is to generate a family or a set of clusters $\mathcal{F}$ which localize Input-Output paths to be on-chip. The set of clusters $\mathcal{F}$ is composed of:

1. A cluster for each multi-fanout net which has a unque set of outputs.
2. A cluster for each cone of logic corresponding to each node on the OSET.

Figure 1 shows a cluster associated with multi-fanout net *x*. The cluster is composed of cones associated with nodes o_1, o_2 and o_3. The cluster has an olist $\{o_1, o_2, o_3\}$ and an input list $\{i_1, i_2, \ldots, i_k\}$. The complexity of the cluster generation phase is $\bigcirc(V^2)$.

5 Covering Phase

The goal of the covering phase is: Given a circuit $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, a family of clusters $\mathcal{F} = \{f_1, f_2, \ldots, f_n\}$, and an FPGA device $\mathcal{D}$ into which the network is to be partitioned, construct a cover $\mathcal{C} \subseteq \mathcal{F}$ of $\mathbf{G} = (\mathbf{V}, \mathbf{E})$. We are interested in selecting a cover $\mathcal{C}$ in which:

1. Each $b_i \in \mathcal{C}$ is feasible (i.e. fits in FPGA device $\mathcal{D}$).
2. Number of chip crossings on each circuit path from the **ISET** to the **OSET** is minimized.
3. Replication of logic is minimized.

The replication of logic may become necessary in order to avoid path cuts when clusters in the cover share logic and must get placed in different chips due to area or io constraints. An important goal of the covering phase is to minimize the replicated area in order to have minimal impact during the packing phase. It should be noted here that the family of clusters $\mathcal{F}$ may contain infeasible clusters (i.e. clusters that can't fit into $\mathcal{D}$) due to violation of area or i/o constraints or both. If an infeasible cluster b_i is chosen to be in the cover $\mathcal{C}$, then it becomes necessary to partition b_i into $\{b_{i1}, b_{i2}, \ldots, b_{ir}\}$, where each b_{ik} is feasible. The feasible partition of b_i (i.e. $\{b_{i1}, b_{i2}, \ldots, b_{ir}\}$) is placed on the cover $\mathcal{C}$.

In order to limit the path cuts to only those cones that are infeasible, we break the input graph **G** into two parts called the *feasible graph* $\mathbf{G_f}$ consisting of all feasible cones in the **OSET**, and the *infeasible graph* $\mathbf{G_i}$ consisting of infeasible cones in the **OSET**. The covering problem is then solved separately on the feasible graph $\mathbf{G_f}$ and the infeasible graph $\mathbf{G_i}$ to produce covers $\mathcal{C}_f$ and $\mathcal{C}_i$ respectively. The cover for the starting network **G** is constructed as: $\mathcal{C} = \mathcal{C}_f \bigcup \mathcal{C}_i$. Note that the graph $\mathbf{G_i}$ is empty if there are no infeasible cones in the original circuit **G**.

5.1 Set Cover for Feasible Graph G_f

The Set Covering Problem (SCP) is an important optimization problem that models many resource-selection problems. The SCP is known to be NP-Hard, but, good approximation algorithms exist [4]. In this paper we use a variant of [4] for set covering. While the classical SCP attemps to find a minimum cardinality cover, our goal is to produce a cover that has the maximum benefit, where benefit of a cluster is defined as a function of area, ios, and outputs covered by the cones of the cluster. The pseudocode outlining our approach for set-covering is shown in figure 2.

```
Set_Cover(G_f)
1.    F_f = Generate_Clusters(G_f);
```

2. For each $f_i \in \mathcal{F}_f$
3. If f_i is feasible
   $f_i.score = \alpha \times$ area_benefit +
   $\beta \times$ inp_benefit +
   $\gamma \times$ num_cones_covered;
   else
   $f_i.score = 0$;
4. $\mathcal{C}_f = \phi$
5. While ($\mathcal{C}_f$ not a cover of G_f)
6. Select a cluster **bc** with highest $f_i.score$;
7. $\mathcal{C}_f = \mathcal{C}_f \bigcup \{bc\}$;
8. Update Cover;
9. Update Affected Clusters;

Figure 2. Pseudo-Code for Set Cover of G_f

In the first step we generate all the unique clusters of the graph. In step 2 a score is assigned to each feasible cluster f_i in $\mathcal{F}_f$. The term *area_benefit* represents the number of LUTs shared among the cones of the cluster corresponding to its olist. It is computed as follows from the individual cone areas and the cluster area:

$$area\_benefit = \sum_{j=1}^{|f_i.olist|} conearea(f_i.olist_j) - f_i.area$$

This shared area results from the common sub-expressions and common cubes shared by the cones in $f_i.olist$. This term plays an important role in minimizing the amount of logic that is replicated. A larger area_benefit in general implies lesser area and input duplication costs.

The term *inp_benefit* results from the inputs that are shared among the cones of the cluster, and is computed as follows:

$$inp\_benefit = \sum_{j=1}^{|f_i.olist|} inputs(f_i.olist_j) - |f_i.ilist|$$

Larger input sharing in general retains heavily I/O related cones together, minimizing I/O usage among the chips of the partition. It may also indirectly influence better packing.

The term *num_cones_covered* is the number of outputs on $f_i.olist$. Larger the number of cones covered implies a smaller cover size $\mathcal{C}_f$ and a relatively simpler packing problem. The coefficients α, β, and γ are chosen to reflect the importance of area_benefit, followed by inp_benefit and lastly the num_cones_covered.

The set covering algorithm is greedy. The best feasible cluster **bc** which represents the best area, io savings, and the maximum number of cones covered is selected in each iteration. The selected cluster **bc** is then placed on the cover $\mathcal{C}_f$. As a result of selecting **bc** a number of clusters may have to be updated. The updating process (step 9) involves eliminating clusters on the fanout of the selected cluster and also updating other clusters in the fanin cones of the selected cluster's outputs. As a result of this update (step 9) several clusters that were infeasible earlier may now become feasible. Since the update step invoves depth-first search of its fanout cone (which is sparse), and depth first searches for potentially $\bigcirc(V)$ clusters (which are sparse) encountered in the fanin cones of **bc's** output list, its complexity is $\bigcirc(V^2)$. The overall complexity of the set cover approach is $\bigcirc(|\mathbf{OSET_f}| \times \mathbf{V^2})$, where $|\mathbf{OSET_f}|$ is the number of primary outputs and register inputs in the graph $\mathbf{G_f}$.

An example of set covering for a feasible graph is shown in figure 3a. The network consists of $\mathbf{OSET} = \{o_1, o_2, o_3, o_4\}$. The family of clusters $\mathcal{F}$ consists of clusters associated with nets c_1, c_2 and c_3 and also the clusters corresponding to the cones o_1, o_2, o_3 and o_4 of the **OSET**. The cluster associated with net c_1 (referred to as cluster c_1) has an olist $\{o_1, o_2\}$ and ilist I_1, cluster c_2 has an olist $\{o_3, o_4\}$ and ilist I_2, and cluster c_3 has an olist $\{o_2, o_3\}$ and olist I_3. During the set-covering process, cluster c_1 which has the highest score is selected first, followed by the selection of cluster c_2. At this point the cover $\mathcal{C} = \{c_1, c_2\}$ is generated (see figure 3b). Note, that the cover $\mathcal{C}$ confines all paths to be on-chip, and achieves this at the cost of duplicating the fanin cone associated with net c_3. It should be noted that the greedy nature of the covering routine in general promotes exploiting clusters that are strongly related (i.e. c_1, c_2), confining the duplication to relatively smaller portions of logic.

5.2 Set Cover for Infeasible Graph G_i

The property of the infeasible graph is that its OSET contains infeasible cones of logic due to area or i/o violations or both. It is therefore, necessary to partition these cones such that the number of chip crossings on any path is held to a minimum.

The set covering approach used here is similar to SCP algorithm proposed for the feasible graph G_f seen earlier. Since each chosen cluster **bc** is infeasible, it is further partitioned into feasible clusters with minimum number of chip crossings on any circuit path from the cluster's input to the cluster's output. The minimum path cut partitioning is achieved by recursively bi-partitioning the chosen cluster **bc** into two sets called the divisor set **D** and the quotient set **Q**. The bi-partitioner uses a net-based approach and guarantees that the individual cones in the divisor set are feasible (i.e. fit into FPGA device) and there is at most a single chip crossing on any circuit path from the cluster's input to the cluster's output [5]. This process is terminated when the quotient set **Q** has cones that are feasible (i.e. fit into specified FPGA). The overall

complexity of the set covering problem for the infeasible graph is $\bigcirc(\mathbf{V^2})$.

6 Packing

In this phase we are given a cover $\mathcal{C}$ of the original network $G = (V, E)$, and an FPGA device $\mathcal{D}$ into which the clusters in $\mathcal{C}$ must be packed. The goal is to produce a packing with the fewest number of chips. We perform this packing using a best fit decresing bin packing heuristic. The overall complexity of this phase is $\bigcirc(|\mathcal{C}| \times V)$.

7 Results

We have conducted experiments on several MCNC and ISCAS benchmark circuits. Our experimental set up is shown in figure 4. Each benchmark circuit was first optimized for area with a standard script using misII. The optimized circuit was then mapped to Xilinx FPGAs (4005H, and 3090) using TechMap [5] prior to being partitioned using **SCP** and **ANN** partitioners. The **ANN** partitioner is simulated annealing based and uses a recursive bi-partitioning technique to implement multi-way partitioning that minimizes the number of nets cut and also the number of chips.

In order to determine the quality of the partitioning solutions produced by **SCP** and **ANN** and compare them, we define the following:

1. $Idl\#chips = Max\left(\left\lceil \frac{Network\_IOs}{FPGA\_IOs} \right\rceil, \left\lceil \frac{Network\_Area}{FPGA\_Area} \right\rceil\right)$
   Here, the terms *Network_Area* and *Network_IOs* refer to the number of LUTs and the number of IOs ($PIs + POs$) in the unpartitioned network. The *Idl#chips* serves as a lower bound on the number of chips required to partition the original network. The lower bound is computed by considering the number of LUTs and number of IOs of the given network independently. It should be noted that this lower bound may not be achievable in reality (i.e. $optimal > lower\_bound$).

2. $Areacost = \frac{\sum_{i=1}^{k} chiparea_i}{Network\_Area}$
   The term *Areacost* represents the cost due to logic duplication.

3. $Pincost = \frac{\sum_{i=1}^{k} chipios_i}{Network\_IOs}$
   The term *Pincost* represents the cost due to I/O duplication and internal signals that become I/Os due to partitioning.

circuit	*IOs*	*LUTs*	*Regs*	*Levels*
rot	242	289	0	9
apex6	242	346	0	6
c2670	372	314	0	8
c5315	301	659	0	11
DES	501	1380	0	8
c3540	72	420	0	14
c6288	64	768	0	26
s5378	84	588	164	7
s9234	75	787	211	16
DALU	91	676	0	13
trial	476	635	0	9
trial1	777	1278	0	11
trial2	848	955	0	9
c7552a	314	678	0	10
s13207	194	886	485	10
s15850	151	1325	522	16
s38417	135	4111	1536	13
s38584	322	4370	1394	12

Table 1. Benchmarks Mapped on Xilinx 4005H FPGA.

In table 1 the characteristics of optimized and technology mapped benchmark circuits that serve as starting networks to be partitioned on to Xilinx XC4005H FPGA are shown.

The XC4005H device has 192 I/Os, 392 4-input LUTs, and 392 Flip-Flops. The result of partitioning the circuits shown in Table 1 are shown in Table 2. The term *#cp* refers to the number of chips in the partition. The term *#cc* refers to the maximum number of chip crossings on any circuit path. The terms *pc* and *ac* respectively refer to the pincost and areacosts that were defined earlier.

From the data in Table 2 we observe that the **SCP** partitioner in comparison with **ANN** produces partitions that have 14% fewer chips, 28% fewer pins, and 93% fewer number of chip crossings on each circuit path. This performance and efficient packing was achieved at the cost of duplicating 13% of the logic on the average. It should be noted however that despite the duplication costs **SCP** has fewer number of chips and pins on average. We observe that **SCP** achieves an average chip area utilization of 76% and average chip I/O utilization of 82% compared to **ANN** which achieves average chip area and I/O utilizations of 57% and 94% respectively. As a consequence, it can be seen that **ANN** saturates chip I/Os faster than **SCP** and leads to more number of chips in the partition. We further note that all of the circuits partitioned by **SCP** were successfully placed and routed using Xilinx's PPR. Additionally, we observe that **SCP** matches ideal in 10 test cases, 1 more than ideal in 6 test cases, and 2 more than ideal in the other 2 test cases.

In addition to the potential timing benefits due to maximization of on-chip paths, we believe that smaller pincosts (avg. 28%) and smaller partition sizes (avg. 14%) in general imply easier wiring at the PCB or the MCM levels

and the potential to achieve compact placements, smaller wirelengths, and consequently better overall performance.

We have also mapped and partitioned the benchmarks of Table 1 on to Xilinx XC3090 FPGA. The XC3090 has 320 CLBs and 144 IOs. In comparison with the **ANN** partitioner the **SCP** partitioner achieved 23% fewer chips, 16% fewer pins, 97% fewer path-cuts, at a cost of 14% more area. The results for XC3090 show a similar trend to those discussed earlier for the XC4005H.

The run-times for the individual test cases for partitioning using **SCP** ranged from a few seconds to approximately 20 minutes (for s38417) on Dec-Station 3100.

8 Conclusions

In this paper we proposed a constructive approach to multi-way Partitioning for minimum delay for Look-Up Table Based FPGAs. Our approach to clustering and set covering has a complexity of $\bigcirc(\mathbf{V^2})$, and achieves excellent area/IO tradeoffs to produce a cover of the original network with minimum number of path-cuts on each I/O path of the circuit and minimum replication of logic. Our results have demonstrated the efficacy of the proposed approach and its ability to produce near-optimal solutions.

9 Acknowledgements

This work was performed under NSF contract MIP-9112930. The authors gratefully acknowledge the Resident Study Grant recieved from IBM Corporation.

circuit	*SCP*				*ANN*			*Ideal*
	#cp	#cc	pc	ac	#cp	#cc	pc	#cp
rot	2	0	1.10	1.00	2	1	1.08	2
apex6	2	0	1.17	1.00	2	1	1.24	2
C2670	2	0	1.02	1.00	2	2	1.40	2
C5315	3	0	1.44	1.36	3	4	1.83	2
DES	4	0	1.44	1.01	7	4	2.50	4
C3540	2	0	1.54	1.39	2	7	3.79	2
C6288	3	0	5.39	1.45	3	9	6.42	2
s5378	2	1	3.25	1.07	3	5	5.23	2
s9234	3	1	5.60	1.19	4	7	10.25	3
DALU	3	0	2.40	1.49	3	6	4.74	2
trial	3	0	1.15	1.00	4	3	1.42	3
trial1	5	0	1.20	1.12	6	4	1.50	5
trial2	5	0	1.10	1.00	5	3	1.31	5
C7552a	3	0	1.51	1.10	3	3	1.86	2
s13207	4	1	4.01	1.00	4	9	4.18	3
s15850	5	1	5.78	1.08	6	10	6.15	4
s38417	13	1	16.22	1.14	16	10	21.91	11
s35854	14	1	7.55	1.04	16	7	10.60	12

Table 2. Comparison of SCP with ANN and Ideal for circuits mapped on XC4005H

References

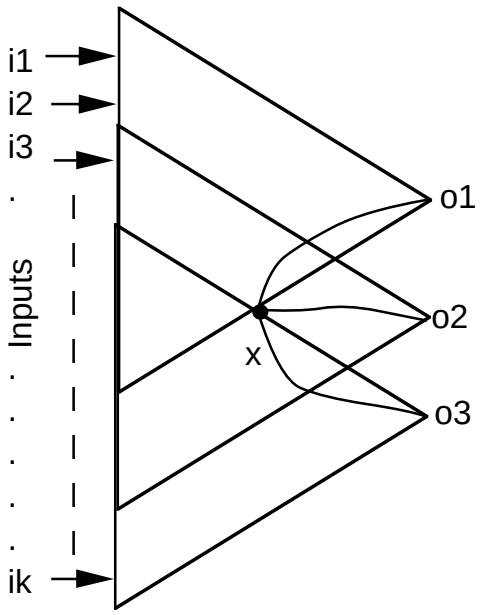
[1] R. Kuznar, F. Brglez, and K. Kozminski, "Cost Minimization of Partitions into Multiple Devices," DAC-93.

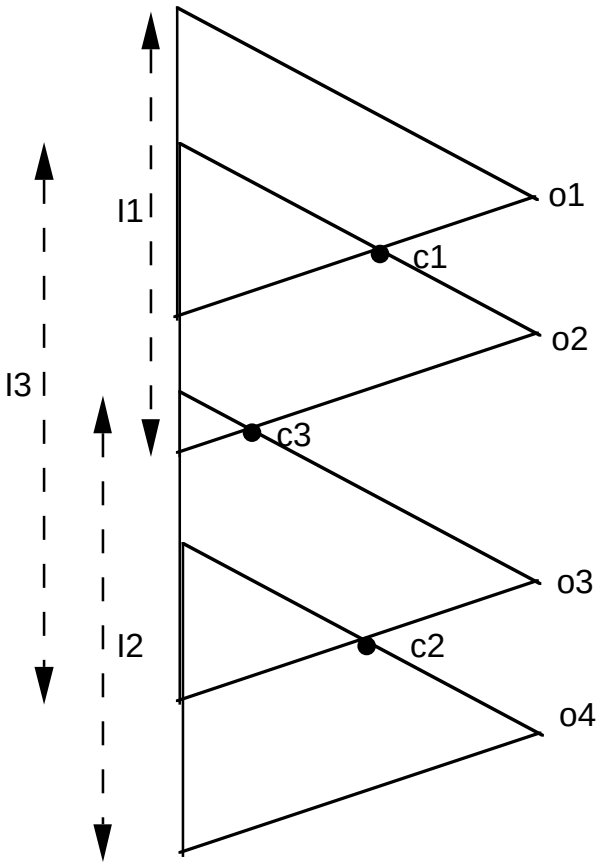
[2] R. Kuznar, F. Brglez, and B. Zajc, "Multi-way Netlist Partitioning into Heterogeneous FPGAs and Minimization of Total Device Cost and Interconnect," DAC-94.

[3] K. Roy and C. Sechen, "A Timing Driven N-Way Chip and Multi-Chip Partitioner," ICCAD-93.

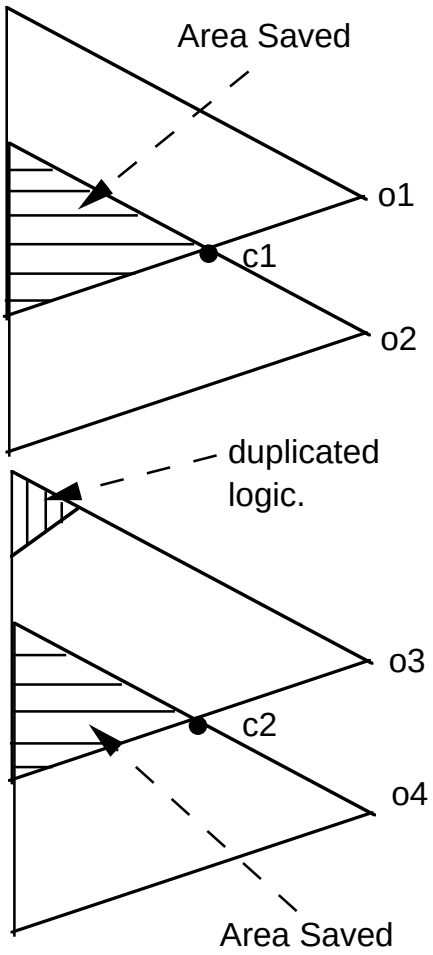
[4] V. Chvatal, "A Greedy Heuristic For the Set-Covering Problem," Mathematics of Operations Research, 4(3), pp 233-235, 1979.

[5] P. S. Sawkar, "Performance Optimized Partitioning and Technology Mapping For FPGAs," Ph.D. Thesis in preparation, Carnegie Mellon University, March 1995.

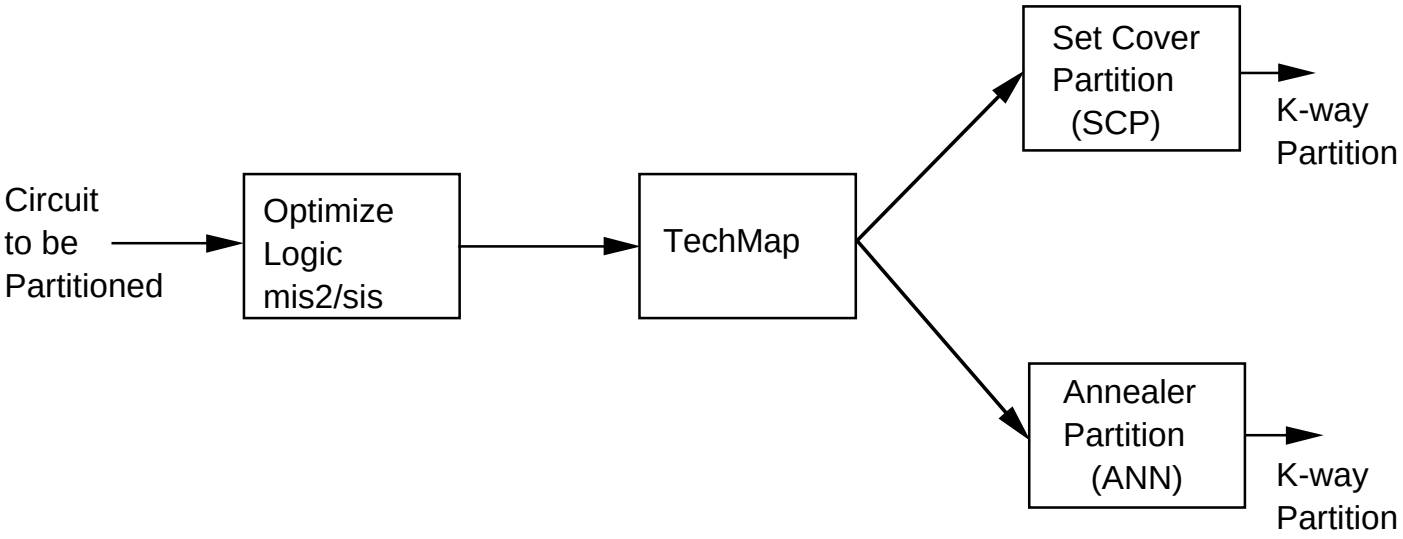

i1
i2
i3
.
Inputs
.
.
.
.
ik
o1
o2
o3
x



a) Original Network and Clusters



b) Covering of Network by c1 and c2.


Circuit
to be
Partitioned
Optimize
Logic
mis2/sis
TechMap
Set Cover
Partition
(SCP)
K-way
Partition
Annealer
Partition
(ANN)
K-way
Partition