CS577 C-based VLSI Design

Project Report -THE RISERS

Team Members:

Guruprasad Sajjan -224161015

Sarwan Dubey- 224161012

Abhishek Khajuria-224161011

K Ajay Kumar Reddy- 224161017

S. Vasanth Kumar- 224161010

Akshay Vitthal Parit- 224161013

Introduction:

Stochastic Search:

Stochastic search algorithms are a class of optimization algorithms that use random sampling to find the optimal solution from a large set of possible solutions.

Logic Locking:

The basic idea behind logic locking is to add additional circuitry to the original design that can be used to control the functionality of the IC. This additional circuitry is usually implemented as a key-controlled gate, which is a set of gates that are controlled by a secret key. The original design logic is modified to incorporate these key-controlled gates, and the key is used to control the output of these gates, effectively locking, or unlocking the circuit functionality.

Approach: Stochastic Search on Logic Locking

Applying stochastic search to logic locking involves using randomized algorithms to explore the search space of possible key assignments in order to find a solution that helps in decrypting the security keys from the source code encrypted using logic locking.

Here are the general steps to apply stochastic search to logic locking:

1. **Define the search space**: Identify the parameters or variables that can be randomized in the logic locking technique, such as the key assignments, and input variables. Define the search space, which is the set of all possible values or combinations of these parameters.

2. **Formulate the optimization problem**: Define the objective function that measures the effectiveness of a particular key assignment or modification in unlocking the logic locked code. This could be based on metrics such as the output of the code, the similarity of the output to expected results, or other measures of correctness.

3. **Generate initial solutions**: Generate an initial set of key assignments or modifications by randomly assigning values to the keys in the search space.

4. **Evaluate solutions**: Evaluate the performance of each key assignment applying it to the logic locked code and measuring its effectiveness using the objective function.

5. **Generate new solutions**: Apply a stochastic search operator, such as mutation, crossover, or random sampling, to the existing key assignments or modifications to generate new solutions. For example, randomly perturb the key values, swap key assignments, or apply other random modifications.

6. **Update the solution set:** Evaluate the effectiveness of the novel solutions using the objective function and update the solution set by selecting the best solutions based on their performance.

7. **Termination criterion**: Define a termination criterion that specifies when the search process should stop where the resource utilization is maximum. This could be a maximum number of iterations, a threshold performance.

- Initialize Population S_0 ($|S_0| = m$) in Search Space (X_{min} , X_{max})
- Evaluate Y_0 for Solution Population S_0
- FOR $t = 0 \rightarrow T_{max}$
  - Compute Fitness Scores P_t from Y_t
  - IF ContinueExploitation = TRUE
    - Generate Children Population S_{t+1}^{RW} and Evaluate Y_{t+1}^{RW}
    - $\{S_{t+1}, Y_{t+1}\} = \text{BEST}_\text{m}\{ S_t \cup S_{t+1}^{RW}\}$
  - ELSE
    - Generate Children Population $\{S_{t+1}^{RC}, S_{t+1}^{RRI}\}$ and Evaluate $\{Y_{t+1}^{RC}, Y_{t+1}^{RRI}\}$
    - $\{S_{t+1}, Y_{t+1}\} = \text{BEST}_\text{m}\{ S_t \cup S_{t+1}^{RW} \cup S_{t+1}^{RRI}\}$
- END FOR
- Best Solution: $\text{BEST}\{S_{T_{max}}, Y_{T_{max}}\}$

ref: slide taken from Dr.Prithwijit Guha ML slides

Stochastic Search Formulation to our problem:

1. Search Space:

Integer Key 1-Key 4: Each of the keys can take values from 0 to 31, inclusive, as it is an integer with a range of 32 possible values.

Boolean Key-5 to Key 7: Each key can take two possible values: either True (1) or False (0), as it is a Boolean key.

So, the overall search space for the 7 keys would be a combination of 4 integer keys each having 32 possible values (0 to 31), and 3 Boolean keys each having 2 possible values (True or False), resulting in a total of 32^4 * 2^3 = 2,097,152 possible key combinations. This represents the space of all possible key assignments that can be explored during the stochastic search process.

2. **The Optimization problem:**

Here we consider the optimization function(cost function):

$$f(key) = || O(key) - O(a.out) ||_1$$

where $|| ||_1$ represents the distance metric which is L-1 norm used, and f(key) represents the optimization function that measures the L-1 norm between the two outputs.

The goal of the optimization problem is to find the key assignment that minimizes the objective function, i.e., the key assignment that produces an output from the obfuscated file closest to the output from the black box executable file (a.out).

Generic approach of Stochastic search will be maximization of objective function, Henceforth, to customize the stochastic search for the existing problem we have to take the negative value of cost function find the solution where the difference is exactly 0.

3. **Generate initial solutions:**

- Generate random values for the 4 integer keys: Randomly assign values from 0 to 31 (inclusive) for each of the 4 integer keys in the key assignment. This can be done using a random number generator function or by random sampling from a uniform distribution.

- Generate random values for the 3 Boolean keys: Randomly assign True or False values for each of the 3 Boolean keys in the key assignment. This can also be done using a random number generator function or by random sampling from a binary distribution.
- Combine the generated values for all 7 keys: Combine the randomly generated values for the 4 integer keys and 3 Boolean keys to form the initial key assignment. This key assignment represents the initial solution for the stochastic search optimization problem.

For example, a possible initial solution (key assignment) could be:

Integer Key 1: 15

Integer Key 2: 8

Integer Key 3: 23

Integer Key 4: 2

Boolean Key 1: True

Boolean Key 2: False

Boolean Key 3: True

Now the first key is [15,8,23,2, True, False, True], each key of random elements is generated to form a list of pop-size number of key sets.

4. **Evaluate solutions:**

The evaluation function would compute the absolute difference between the actual output and the obfuscated output, elementwise, and then sum up these absolute differences to obtain the objective function value.

For example, using the given example output lists:

Actual Output: [35, 190, 70, 24, 196, 199, 197, 194, 199, 154, 66, 93, 225, 26, 4, 58]

Obfuscated Output: [50, 67, 246, 168, 136, 90, 48, 141, 49, 49, 152, 162, 224, 55, 7, 52]

The evaluation function would compute the absolute difference between these two outputs, elementwise, and then sum up these absolute differences.

Compute the element-wise absolute difference between the two output lists:

Absolute Difference = [abs(50 - 35), abs(67 - 190), abs(246 - 70), abs(168 - 24), abs(136 - 196), abs(90 - 199), abs(48 - 197), abs(141 - 194), abs(49 - 199), abs(49 - 154), abs(152 - 66), abs(162 - 93), abs(224 - 225), abs(55 - 26), abs(7 - 4), abs(52 - 58)]

Absolute Difference = [15, 123, 176, 144, 60, 109, 149, 53, 150, 105, 86, 69, 1, 29, 3, 6]

Sum up the absolute differences:

Sum of Absolute Differences = 15 + 123 + 176 + 144 + 60 + 109 + 149 + 53 + 150 + 105 + 86 + 69 + 1 + 29 + 3 + 6

Sum of Absolute Differences = 1033.

Now we add all the possible Sum of Absolute differences for the sets of keys and return the negative of the values in a list.

5. **Generate new solutions:**

- Random Walk:

  In this step, calculate the number of children selection from each candidate(parent) based on their fitness scores, we subtract the minimum fitness score from all the fitness scores and divide by the sum of all the fitness scores. Here are the steps:

  Calculate the number of children selected from each parent based on their fitness scores. We can do this by taking the negative of the fitness scores and subtracting the minimum fitness score to ensure all values are non-negative, and then normalizing them by dividing by the sum of all the fitness scores using the formula:

  Number of children from parent i = [ceil ([(fitness_i – fitness_min) / (sum of all fitness scores)]) for each i].

- **Random Re-Initialization:**

  Here we generate a few random solutions to introduce diversity in the population of solutions and escape local optima. This can be directly done as we did for generation of initial solutions.

- **Random Linear Combination:**
  1. Iterate over each solution in the current population.
  2. Generate a random value to determine whether to swap integer keys or Boolean keys.
  3. If integer keys are selected, randomly select two indices from the first four keys.
  4. If Boolean keys are selected, randomly select two indices from the last three keys.
  5. Swap the values at the randomly selected indices within the solution.
  6. Repeat steps 2-5 for all solutions in the population.
  7. Return the updated population with randomly swapped keys.

  Random linear combination introduces randomness and exploration in the population by swapping values of two keys within each solution. This can potentially lead to the discovery of better solutions by exploring different combinations of keys.

6. **Update the solution set:**

Select the top pop-size number of solutions with the highest fitness scores to form the next generation of the population.

7. **Termination criterion**:

The termination condition for this stochastic search process is defined based on the number of iterations or when the absolute difference between the outputs of the obfuscated code and the black box executable file becomes zero. This indicates that a solution has been found that breaks the keys in the logic-locked code, and further iterations are not necessary.

Keys found from stochastic Search:

Key1: 9

Key2: 15

Key3: 7

Key4: 8

Key5: False

Key6: False

Key7: False