IIT JODHPUR

Major Examination: EEL71090 ML in VLSI CAD [OPEN-BOOK] (Apr'25)

Guidelines (Total time: 180 minutes, Maximum Marks: 40):

- Please read the question paper very carefully. This is an OPEN-BOOK, CLOSED-INTERNET exam.
- **NO clarification is required in any question. If you ask the invigilator any question, -5 marks penalty would be awarded.** In case of any doubt, assume whatever you wish to and state that in your answer. Step-wise marks would be awarded wherever applicable.
- **Attempt any FOUR questions.** All questions carry 10 marks.
- All offline students are asked to submit python programs on GC within 15 minutes of the ending of the exam (Internet access is ALLOWED only after the ending of the exam).

---

1. The goal of this question is to analyze the simulation trace of a design and prepare inputs for a machine learning(ML)-based analysis for design verification. In ML paradigm, large amount of data is crunched into smaller numbers called "features". So, we need features of debug data to begin with. Let's try to understand how we can develop some useful features from the simulation trace of a design. We want to analyze the cycle-accurate behavior of the design in terms of the contents of all flip-flops in the design. As flip-flops are the sequential elements in the design, it makes some sense to analyze the value stored in these flip-flops in each clock cycle (obviously, not infinitely! but over a certain number of clock cycles only). Consider that the design contains M flip-flops and the simulation trace is N clock cycles long. So, we have input in the form of a table with N rows and M columns.

a1	a2	..	..	..	..	..	b3	b9	p1	p2	p3	p4	..	q7
1	0	1	1	0	1	1	0	1	1	0	1	0	1	1
0	..	..	..	..	..	..	..	..	..	..	..	..	..	0
1	..	..	..	..	..	..	..	..	..	..	..	..	..	1
1	0	..	..	..	..	..	..	..	..	..	..	..	..	1
1	..	..	..	..	..	..	..	..	..	..	..	..	..	0
0	..	..	..	..	..	..	..	..	..	..	..	..	..	0
0	1	..	..	..	..	..	..	..	..	..	..	..	..	1

(a) Devise a methodology of deriving features from the simulation trace data as shown above. [6]

(b) Utilize random data (mimicking a trace of 20000 cycles and 1024 flip-flops) and then write a python program to compute the features as mentioned by you in (a)? [4]

2. For Engineering Change Order (ECO), it is important to know that which flip-flop/gate has less complexity so that this could be modified in order to amend (with the least amount of efforts) the design under consideration. One of the ways to track complexity of a flip-flop of the design is through analysis of different logic paths in the respective logic cone. Consider the diagram shown below:

(a)	(b)	(c)
G1 G2 G5 G7 G1 G6 G3 G31 G9 G7 G6 G78 G781 G51 ------- ---------- G45 G789 G432 G47	G1 G2 G5 G7 G1 G21 G51 G678 G7 G1 G24 G53 G781 G7 G1 G22 G55 G76 G890 G7	G1 G2 G5 G7 G1 G21 G51 G678 G7 G1 G24 G51 G678 G781 G7 G1 G22 G55 G76 G890 G7

In above figure, the end-points represent the flip-flops ALWAYS. So, in Fig. (a), flip-flop(s) are G1, G7, G9, G51, G45 and G47 and rest all are gates. To any flip-flop, an incoming path is considered as the linkage from other flip-flop(s). So, in the line "G1 G2 G5 G7" G7 flip-flop has an incoming path from G1. Similarly, "G7 G6 G78 G781 G51" means that G51 has an incoming path from G7. Also, "G45 G789 G432 G47" means that G47 has an incoming path from G45. We can rank the flip-flops of the netlist in the descending order of the total number of incoming path(s) to them. Therefore, by definition of "incoming path" as described above, we may rank these flip-flops as below: Say, Flip-flops are represented as Fi, Fj,

Fk and so on. They respectively have incoming paths as 23, 45, 13 and so on. So, the descending order of ranking would be: Fj, Fi, Fk and so on.

In Fig. (b), even though we have 4 incoming paths to G7 from G1, we would consider only one path from G1 to G7. G7 may have other incoming paths from other flip-flops. Alternatively, we can think as below: From any path from flip-flop Fi to Fj, count the number of gates. Say, this number is N. The value "N" is to be used as a score for that incoming path from Fi to Fj. So, as per the above Figure, in line "G1 G2 G5 G7", it is to be understood that G7 flip-flop has an incoming path from G1 with a score of 2. Similarly, "G7 G6 G78 G781 G51" means that G51 has an incoming path from G7 with a score of 3. Also, "G45 G789 G432 G47" means that G47 has an incoming path from G45 with a score of 2. In the same manner, "G1 G22 G55 G76 G890 G7" has an incoming path from G7 with a score of 4.

In Fig. (c) above, "G51 G678" is occurring together (i.e., occurring as a tuple) in more than one path, i.e., it occurs in 2 paths. We can perform the above gate count-based ranking by considering these two gates occurring together as one.

(a) Devise a technique to rank all the flip-flops of the design (using above methods or supplement with new methodologies) so that a ML model can be developed for predicting the feasibility of changing the logic around a flip-flop during the ECO stage (after base tape-out is done)? [6]

(b) Implement partially or completely the ranking technique devised by you in (a) ? Consider random logic paths for minimum of 512 flip-flops in the design? [4]

3. Refer to Chapter-5 [Section 5.6.2] of "VLSI Physical Design" book by Prof. Andrew B. Kahng and others (uploaded on course GC). Consider the example (determining routability) on page-159 (look for this page number in PDF page-number search box). We wish to develop a ML model so that the routability could be predicted given the connectivity graph, routing region information and routing obstacle constraints as inputs. Note that these routing constraints can not be completely known beforehand, therefore, it makes sense to use ML technqiues for the routability feasibility predictions.

   (a) How can we develop a dataset for training any ML model for achieving net (wire) routability prediction? You can consider the logic netlist with nets (wires) and logic gates information as inputs in the dataset. Also, you can consider the horizontal and vertical routing constraints limits randomly. What information you can consider as output (or label in case of supervised ML) in this dataset? Why? [6]

   (b) Which ML/DL model is best suited for solving this routability prediction problem given the particular dataset devised by you in (a) above? How? [4]

4. Consider the solution proposed by members of Team 8 (page-17) in the paper, "Logic Synthesis Meets Machine Learning: Trading Exactness for Generalization" (uploaded on GC).

   (a) What are the ways to improve the accuracy in designing (i.e., learning) the circuit netlist from the provided input-output relationships as dataset during the model learning process? [3]

   (b) Note that the solutions presented in this paper mostly result in combinational circuits. How can the number of state elements (flip-flops) be fixed in this ML-based design synthesis process? [3]

   (c) How does any neural network (NN)-based learning methodology like ANN/CNN perform in comparison to answer in (a)? Suppose we discard the computational complexity, how NN-based techniques can be enhanced for the purpose of design synthesis? [4]

5. Refer to Chapter-8 [Section 8.3.2] of "VLSI Physical Design" book by Prof. Andrew B. Kahng and others (uploaded on course GC). Consider the example (determining STA into placement) on page-247 (look for this page number in PDF page-number search box).

   (a) How can we prepare a dataset for building a ML model to predict the worst-case negative slack (WNS)? Which ML model may be ideally suited for this objective? [3 + 3]

   (b) Generate data (with random values) provided as per the formulation proposed on page number-247 and check if this dataset is similar to the one you hypothesized in (a) above? [4]
