# IIT JODHPUR

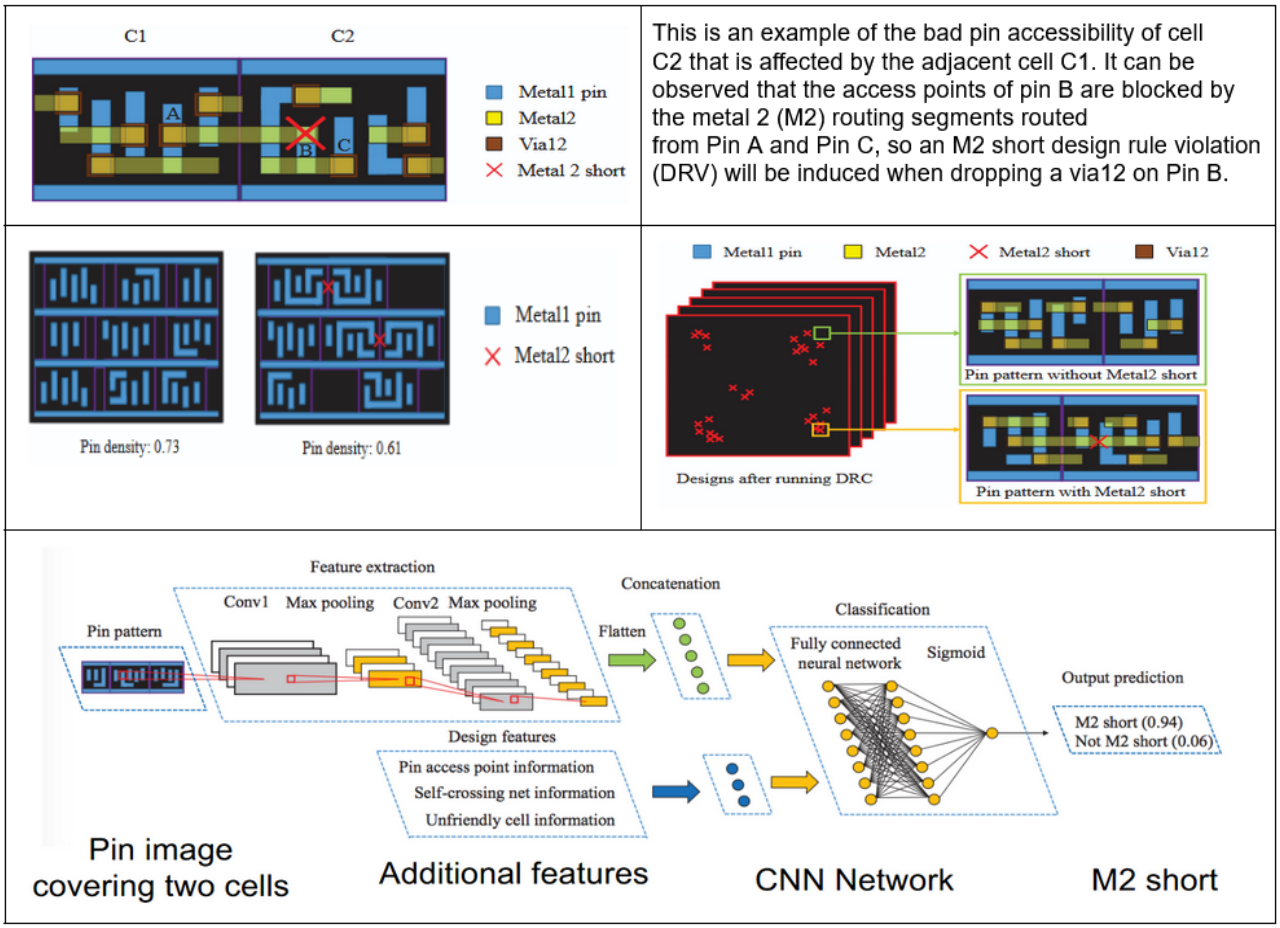
## Major Examination: EEL7698 AI for EDA (Nov'25)

Guidelines (Total time: 3 Hours, Maximum Marks: 40):

- Please read the question paper very carefully. All questions carry equal marks.
- For each question, take all the related assumptions that you may need.

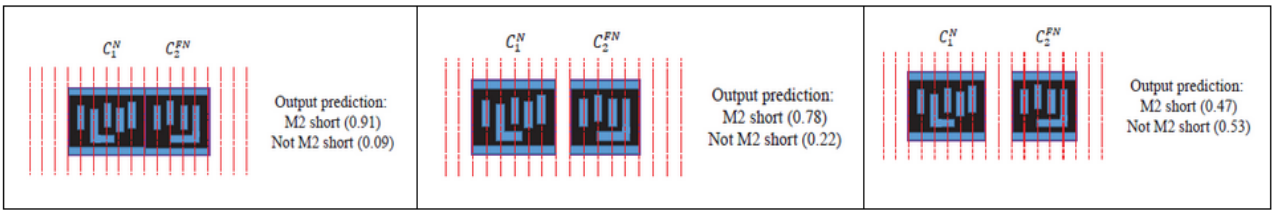
---

1. It is a difficult task to check pin blockages/access points in complicated designs. So, a training based methodology might be useful. Below shows the different parts of such a methodology and one deep learning model applied to the problem formulated as a clarification scenario (short/not).



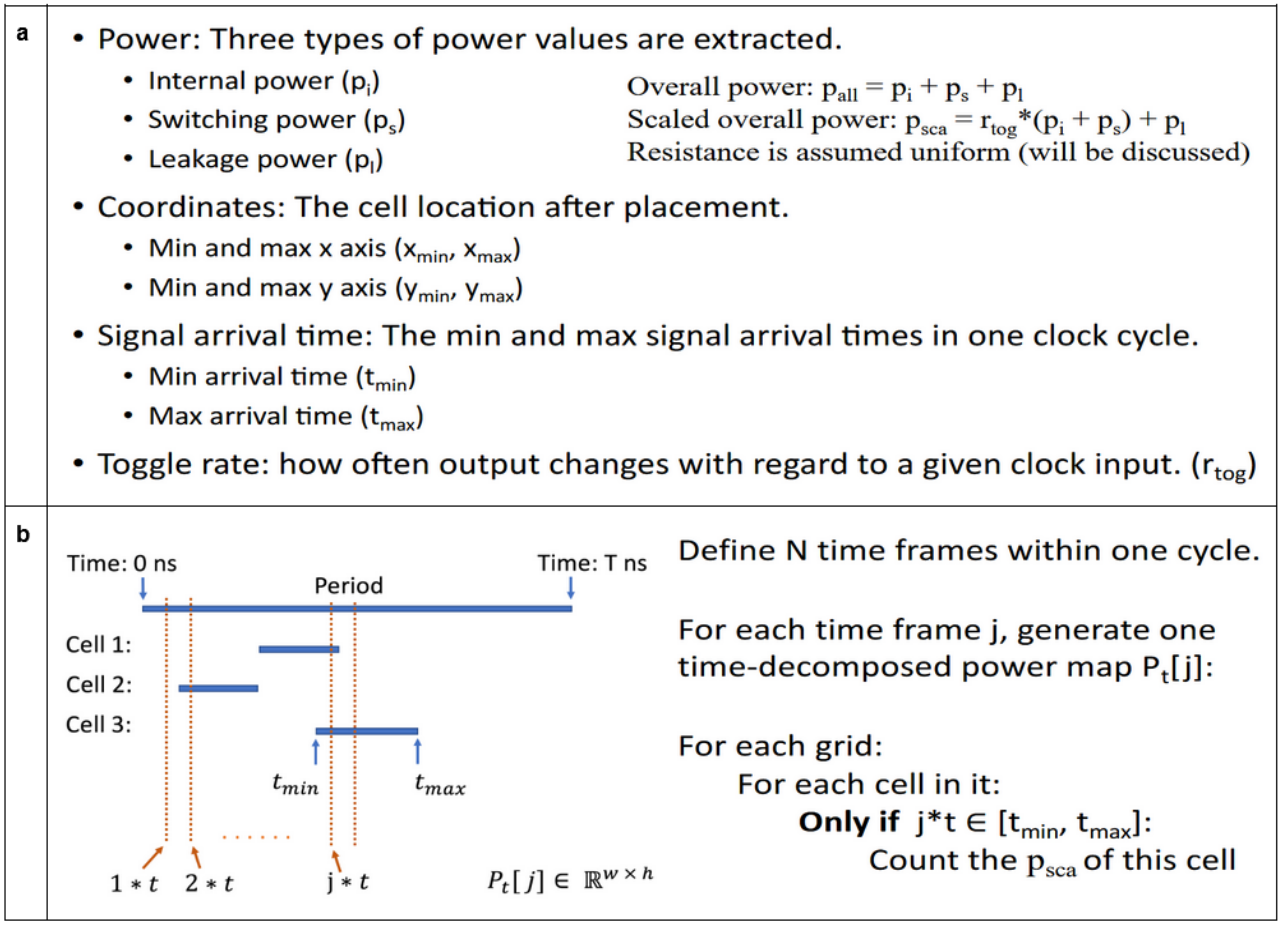

Some definition of related terms (in above figure) are as given as follows

- Pin access point information: This feature is computed by the ratio of the number of pins to the number of access points in the input pin pattern.
- Self-crossing net information: A self-crossing net is caused by the fly lines of two local nets that intersect to each other. A fly line is the straight line connecting the left-bottom access points of two pins. This feature is computed by the ratio of the number of self-crossing nets to the number of local nets in the input pin pattern.
- Unfriendly cell information: With routed designs, we first compute the frequency that each library cell induces M2 shorts. Then, this feature is obtained by the ratio of the the sum of the frequencies of the cells covered by the pin pattern to the total frequency of all cells.




**Examine how the chosen DL model is suitable for this problem? What could be an alternative model for this problem?**

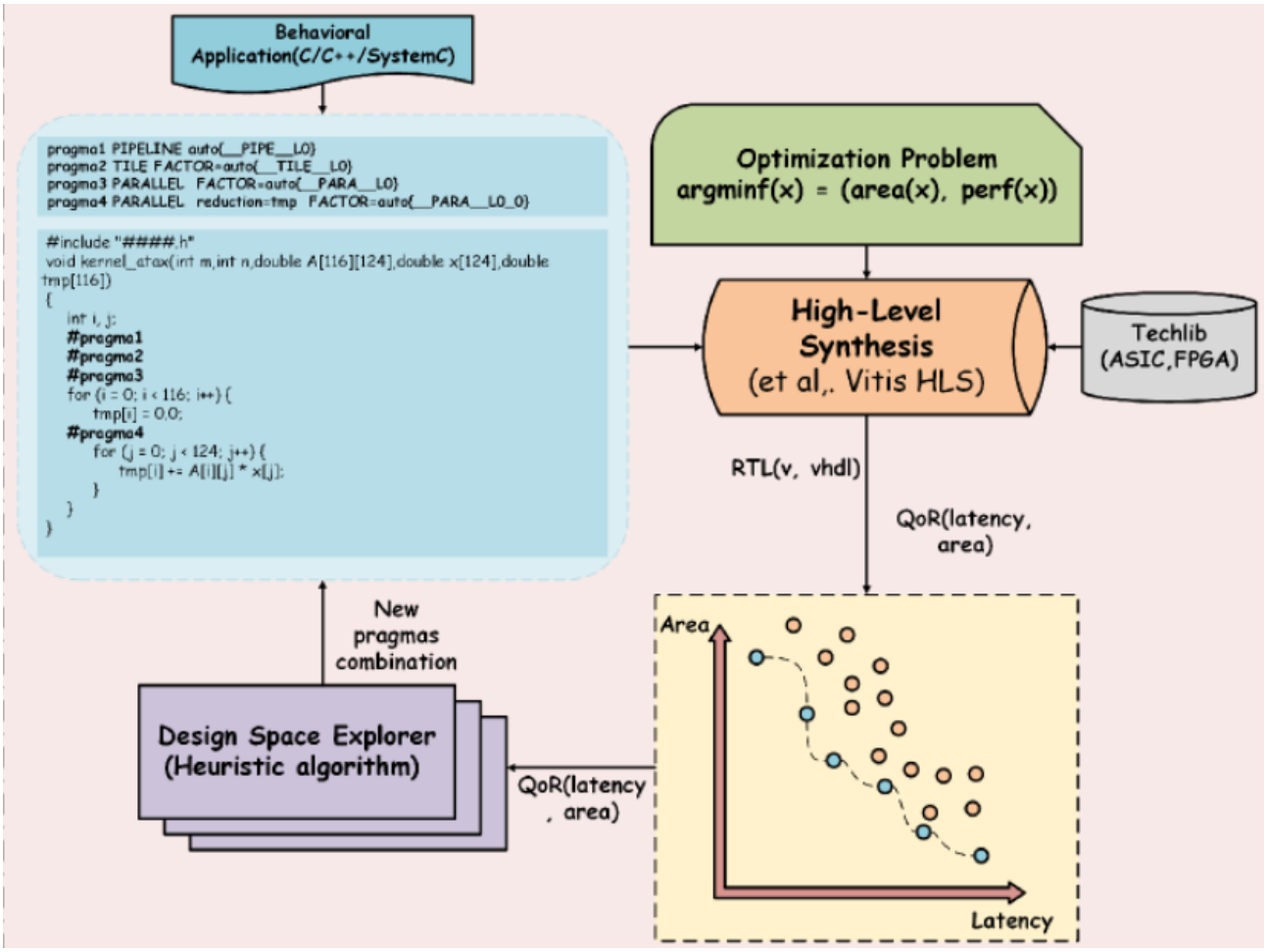
2. Dynamic IR drop is voltage drop at power pin of a standard cell. As estimation of IR drop is a computationally-intensive process, AI-based methods can be better suited for IR drop estimation. For dynamic IR drop, the peak IR drop in the design can be analyzed either using vectorless analysis or vector-based analysis using simulation patterns from value change dump (VCD) files. Vectorless IR drop analysis is highly desirable for IR mitigation during physical design for two main reasons. Firstly, for a large chiplet, vector-based IRdrop analysis requires a huge number of simulation patterns to cover most regions and thus can be unbearably slow. Secondly, designers are unable to obtain accurate power simulation patterns early in the design process. For large industrial designs, multiple teams work on different RTL units in parallel and the overall simulation patterns change throughout the design process. Vectorless IR drop provides a faster and earlier estimation in this case, however, accurate estimation is more difficult than vector-based due to the increased diversity in switching activity distribution.




One methodology of feature extraction of each cell is shown in (a) in the above figure. Therefore, IR drop would be a label in the training process. To capture the worst transient local IR drop,

time decomposition-based flow is outlined in (b) of the below figure. To estimate IR drop, every design is tessellated into an array of tiles, each of which is an l×l square. The tile size l may control the granularity of this solution. A design with the size of W × H is represented as a w × h matrix, where w = W/l and h = H/l. The IR drop at each tile is the mean value of IR drop of all cells within it. Further, space decomposition amortizes cell power into any grid tiles occupied by the cell. We can assume the regular squares are grid tiles and grey rectangles are cells. **Develop AI/ML model to estimate the IR drop using the above problem formulation?**

3. High-Level Synthesis (HLS) has emerged as a modern alternative to traditional RTL-based VLSI design. HLS enables users to leverage high-level programming languages, such as C/C++, for hardware design tasks, offering synthesis options to fine-tune results. Proven in practice, HLS provides numerous advantages in hardware design and has become the mainstream approach, particularly for FPGA implementations. In High-Level Synthesis (HLS) step, Design Space Exploration (DSE) is essential for generating hardware designs that balance performance, power, and area (PPA). To optimize this process, we can employ ML-based methods to predict quality of results (QoR). These predictors serve as evaluators in the DSE process, effectively bypassing the time-consuming estimations traditionally required by HLS tools. In the below figure, the traditional way to carry out DSE is shown below. **To optimize/improve this procedure, devise a ML-based technique to perform DSE in HLS?**

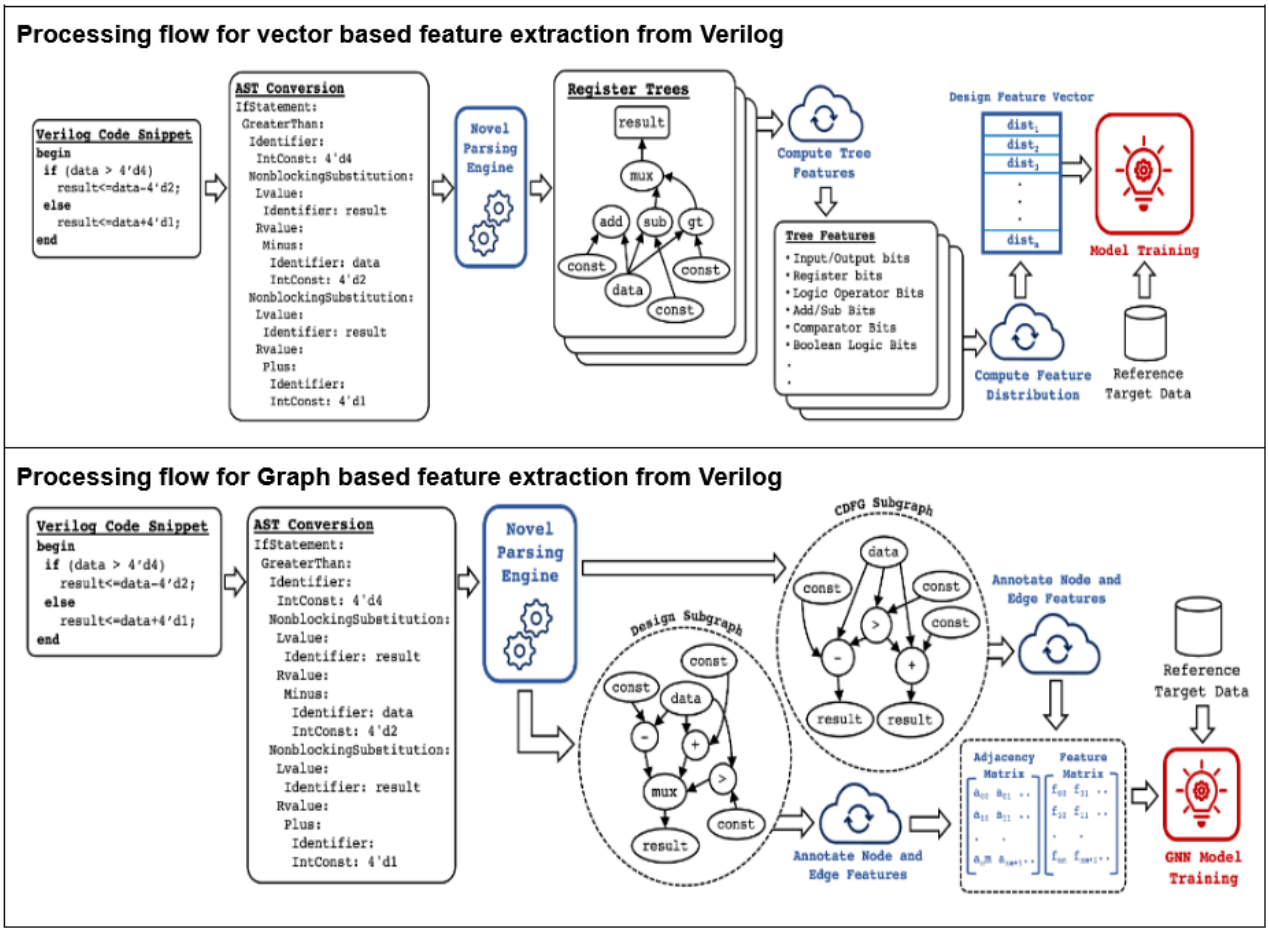



The above workflow begins with C/C++ behavioral descriptions augmented with pragma directives and parameterized optimization targets. These quality of results (QoR) metrics (e.g., latency, resource utilization) of the specifications can be obtained using HLS tools/machine learning models. A design space exploration engine then selects novel pragma combinations through

multi-objective optimization, generating updated behavioral descriptions for subsequent iterations. This cyclic refinement process continues until the convergence to a Pareto-optimal configuration set that optimally balances competing design constraints. Therefore, accurate QoR prediction of each design and efficient design space exploration are the most crucial tasks in HLS DSE.

Typically, four steps are involved here: dataset generation, model training, inference, and DSE. During dataset generation, labels are extracted from both HLS reports (e.g., latency, resource estimates) and implementation reports (e.g., post-place-and-route resource usage, timing closure) to enhance prediction accuracy.

4. In order to have an overall assessment of quality of a RTL (i.e., Verilog code), we need to run the model several times with the desired synthesis recipe parameter values. For assessing key performance metrics such as TNS(Total Negative Slack), area, power etc., we can have Pareto front curve among all results from a Verilog code independent of synthesis recipe parameters. The data collected can be utilized to train a model. Below figures represent the flow for feature extraction from RTL in two different ways.




- Explain with an example, which of the above feature extraction methodology is more suitable?
- After the features are extracted, which DL method is appropriate for building the model?
- Comment on whether the feature extraction method as shown above is complete (i.e., not missing any useful information)?
- What are the chances of on-chip synthesis of the DL method that you have mentioned above so that an online RTL evaluation technique can be implemented?