MAVIREC: ML-Aided Vectored IR-Drop Estimation and Classification

Vidya A. Chhabria[1], Yanqing Zhang[2], Haoxing Ren[2], Ben Keller[2], Brucek Khailany[2], and Sachin S. Sapatnekar[1]
[1]University of Minnesota; [2]NVIDIA Corporation

***Abstract*—Vectored IR drop analysis is a critical step in chip signoff that checks the power integrity of an on-chip power delivery network. Due to the prohibitive runtimes of dynamic IR drop analysis, the large number of test patterns must be whittled down to a small subset of worst-case IR vectors. Unlike the traditional slow heuristic method that select a few vectors with incomplete coverage, MAVIREC uses machine learning techniques—3D convolutions and regression-like layers—for accurately recommending a larger subset of test patterns that exercise worst-case scenarios. In under 30 minutes, MAVIREC profiles 100K-cycle vectors and provides better coverage than a state-of-the-art industrial flow. Further, MAVIREC's IR drop predictor shows 10X speedup with under 4mV RMSE relative to an industrial flow.**

I. Introduction

IR drop analysis estimates the deviation from the ideal supply voltage to meet IR drop constraints across all corners and scenarios. Today, estimating voltage at each node amounts to solving systems of linear equations with billions of variables, with runtimes of several hours on industrial-scale designs even for static analysis. To model switching patterns under real workloads, designers perform dynamic IR drop simulation for several vectors, each spanning a few hundred thousand cycles. Therefore, IR drop analysis in industry is performed for a subset of switching patterns. This subset contains a small number of worst-case *slices* of all vectors, chosen by a vector profiling flow, where a slice corresponds to continuous multicycle time windows of a vector. Fig. 1 (left) shows the industrial flow for vectored IR drop estimation. The blue box performs vector profiling by scanning the entire 100k-cycle vector and recommends a small number of short (up to 20 cycles) worst-case-average-power slices (yellow box). The slices are then sent to the rail analysis engine (orange) to estimate IR drop for each slice. However, such vector profilers are (1) *Slow*: extracting the top three slices of a 3.2 million instance design for a vector of 5000 slices takes two hours; (2) *Approximate*: the heuristics inaccurately sort slices in order of their average power [1] where a low average power does not necessarily imply a low IR drop; (3) *Computationally limited*: the number of recommended slices is limited by the computational bottleneck of rail analysis, and often do not cover all IR-critical regions. Thus, there is a need for a fast and accurate IR drop estimator to profile large sets of vectors and accurately recommend worst-case slices of the full vector.

Prior machine learning (ML)-based approaches [2]–[4] address IR drop analysis runtime, but none are suited for vectored dynamic IR drop analysis. While dynamic vectorless full-chip IR drop analysis is addressed in [3], its inference is slow – our analysis estimates weeks of runtime for full 100k-cycle vector profiling using this. A U-Net based solution in [4] provides faster ML inference than [3] but is targeted at static IR drop. Moreover, [3], [4] provide coarse-grained tile-level IR drops, rather than fine-grained instance-level IR drops, and the single IR value per tile limits transferability (reuse) of the model across designs (Section II-B).

We propose MAVIREC (Fig. 1 (right)): a fast, accurate, and novel vector profiling system that recommends worst-case IR drop switching patterns with high coverage of IR hot spots, using a fast and accurate under-the-hood IR estimator. The estimator uses a novel ML architecture based on U-Nets [5] with key changes: (1) It introduces 3D convolutional layers in the model that capture the sparse temporal switching activities, and (2) It employs a regression-like output layer that enables IR-drop prediction at instance-level granularity. In addition, the use of direct instance-level features as a part of this regression equation enhances transferability and provides model interpretability (see Section II-B). MAVIREC features:

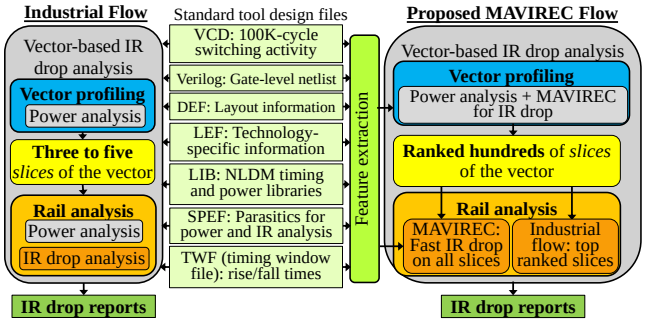

Fig. 1: An industrial flow (left) vs. the MAVIREC flow (right).

- A novel tailored U-Net-based ML model for accurate dynamic IR drop estimation, achieving 10× speedup over industrial flows, at the per-instance (i.e., per-logic-gate) granularity, using 3D convolutional layers for capturing switching activity.
- A novel recommender system that uses fast ML inference to select hundreds of slices compared to conventional flows that select 3–5.
- An accurate and thorough system for profiling long vectors (≈ 100k-cycles), maximizing regional coverage, on industrial scale designs in 30 minutes (4× speedup vs. industrial flow).

This paper highlights a brief overview of MAVIREC with more details in [6].

II. MAVIREC for Vectored Dynamic IR Estimation

MAVIREC's ML inference scheme for rail analysis in Fig. 1 (right), is key for vector profiling. The scheme first performs feature extraction and then uses a trained model for IR drop estimation.

A. Feature extraction

The MAVIREC ML model uses two types of features, differentiated by the method in which they are represented:

- Instance-level power and effective distance to via stack
- Spatial and temporal full-chip tile-based power maps, and spatial time-invariant resistance maps

Table I lists the features used in our ML model, encapsulating power information, the local PDN topology, and switching activity. The feature generation is a two-part process of extracting instance-level features and generating 2D spatial and temporal feature maps.

Instance-level feature generation We run three standard tools to extract instance-level information from the design:

1) a power analysis tool to generate a per-instance switching (p_s), internal (p_i), and leakage power (p_l) report.
2) a DEF reader to extract (a) instance locations (b) effective distance of an instance to power rail via stacks in the immediate (5μm) neighborhood. The effective distance metric is: $r^{-1} = d_1^{-1} + ... + d_V^{-1}$ where d_i is distance from the instance of the i^{th} of V via stacks.
3) a tool that extracts the times when instances toggle, from the VCD file (industry-standard format) for each slice.

TABLE I: MAVIREC's features for dynamic IR prediction where the lowercase symbols are instance-level and uppercase are at the tile-level.

List of all features	
Internal power: p_i, P_i	Overlap power: p_{ol}, P_{ol}
Leakage power: p_l, P_l	Total power: p_{tot}, P_{tot}
Switching power: p_s, P_s	Effective distance: r, R
Toggle rate scaled power: p_r, P_r	Toggle power at each time step: p_t, P_t
Ground truth training labels	
Instance-level IR drop: IR_i, NA (heat map only)	

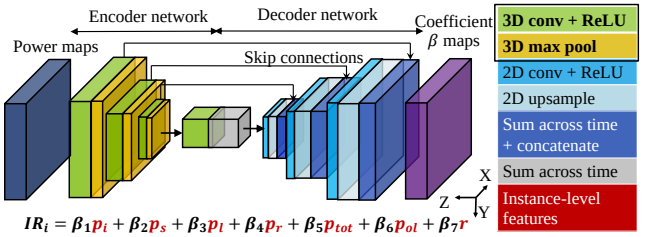

Fig. 2: MAVIREC: 3D convolution and regression model based U-Net architecture for dynamic IR drop prediction.

In addition to p_l, p_s, p_i, and r, there are three other instance-level derived features. The toggle rate scaled and total power are given by $p_r = p_l + \tau_i(p_s + p_i)$ and $p_{tot} = p_l + p_s + p_i$ respectively, where τ_i is the average toggle rate of the instance in the specified slice. The overlap power p_{ol} sums up the p_r for neighboring instances that share a common timing window.

Generation of 2D spatial and temporal maps as features The map creation step performs a spatial and temporal decomposition of the per-instance features similar to [3]. For the spatial-tile based features, we associate the location of each instance with its corresponding power/effective distance attributes to create 2D distributions of these attributes at a fixed granularity (*tile* size). We choose a tile size of $W_t = 2.5\mu$m $\times$ $L_t = 2.5\mu$m; a design of size $W_c \times L_c$ corresponds to an "image" of $W(= W_c/w_t) \times L(= L_c/l_t)$ pixels. For each instance-level feature in Table I, its tile-based counterpart adds per instance power-related features, and takes the maximum effective distance over all instances, in the tile.

For dynamic IR drop analysis, we consider p_t (the power of an instance at each time step) which is generated by dividing the n-cycle time window into $n \times t$ time steps, where n is usually predetermined by the designer as the window size of interest (we use $n = 20$) and t is a hyperparamter to be tuned. It was observed that $t = 5$ provided the best results with MAVIREC. The power of an instance at time step j is given by: $p_t(j) = \sum_{j=1}^{100} p_l + b_j(p_i + p_s)$ where the Boolean variable b_j is 1 only if the instance toggles at time step j. Thus, we create time-decomposed power maps at each time step using the toggle information and timing windows.

B. MAVIREC architecture

Fig. 2 shows the structure of MAVIREC for vectored dynamic IR drop estimation, with layer descriptions provided in the legend. It consists of two subnetworks for (i) encoding (downsampling) and (ii) decoding (upsampling), with skip connections between the networks. This architecture is based on fully convolutional (no fully connected layers) U-Net models [5] which makes them fast and input-image-size independent [4] enabling capture of local and global spatial neighborhood features.

The work in [4] uses U-Nets for estimating *static* IR drop. However, for reasons listed in Section I, this model is not suitable for vectored dynamic IR drop analysis. In addition, typical of real workloads, cells switch at only a few spatial and temporal locations. This makes the temporal power map P_t sparse and difficult to capture in the zero-dominant data. As shown in Section IV-A, merely accounting for these sparse temporal features is not sufficient, and the ML architecture must aid to capture these sparse local changes accurately. There are two key differences from the U-Net in [4] that are crucial for overcoming its limitations:

- use of *3D convolutional layers* (green layers) in the encoding path captures temporal simultaneous switching activity.
- use of a *regression-like layer* at the end of the decoder that incorporates instance-level input features and the instance-level IR drop $\boldsymbol{IR}_i$ (equation in Fig. 2) using $\boldsymbol{\beta_i}$, the coefficient matrix predicted by the U-Net like structure.

3D convolutional layer in the encoder Unlike a 2D convolutional layer, a 3D convolutional layer restricts the number of channels to the specified filter size in the channel dimension, thus considering only a small local window of channels. Due to zero-dominance in the data, a 2D convolutional layer model fails to capture key non-zero toggle activity regions and time steps. Intuitively, a small local window of channels which a 3D convolutional layer considers would better capture simultaneous and sparsely-distributed switching activity(see IV-A). Therefore, MAVIREC uses a $3\times3\times3$ filtered 3D convolutional layer (green layer in Fig. 2) in the encoding path instead of a regular 3×3 2D convolutional layer as in U-Net. MAVIREC has $n \times t + 7$ tile-based channels (Table I):

- $n \times t$ temporal power maps (P_t) feeding the encoder network
- 7 tile-based spatial features (P_i, P_l, P_s, P_r, P_{ol}, P_{tot}, R)

where $n \times t$ represents the number of time steps (Sec. II-A).

MAVIREC consists of three pairs of 3D convolutional layers and max pool layers in the encoder network and a symmetric decoder network. Since the decoding path uses 2D convolutions, the interface between the 3D embedding in the encoder and the 2D embedding in the decoder sums up the embedding along the temporal dimension (dark blue boxes in Fig. 2) through concatenation/skip connections.

Regression-like layer in the decoder For IR drop prediction at a per-instance level, MAVIREC leverages a regression-like layer at the end of the decoder path. The predicted coefficients are based on the $n \times t + 7$ spatial and temporal tile-based channels as input. The coefficients (β_i) predicted for every tile by the U-Net-like structure is multiplied with the per-instance feature values defined in Sec. II-A. This architecture provides three key advantages over [3], [4]:

1) *Improved transferability* compared to prior art, as the model uses both instance-level and tile-level features to predict IR drop per-instance. The instance-based features capture fine-grained variations that is otherwise lost due to the averaging nature of U-Net convolutions. Instead of learning the IR drop values directly as in [3], [4], the U-Net-like structure learns the relationship ($\boldsymbol{\beta_i}$) between the features and the IR drop, which is universal across designs.
2) *Improved instance-level IR drop predictability* compared to prior art, which is useful for a variety of applications such as IR-aware STA and instance-based IR drop mitigation.
3) *Model interpretability* as the predicted coefficients provide information on the weights associated with each feature. The coefficients correspond to feature sensitivity, and allow a designer to assess root causes of an IR drop violation.

A trained MAVIREC model is reusable, without the need for retraining when faced with new designs/vectors for a given technology. Although the prediction is on an instance level granularity, this can be modelled as an element-wise matrix multiplication for scalability.

III. MAVIREC FOR VECTOR PROFILING

Real vector workloads can typically have hundred of thousands of cycles, which corresponds to ∼5000 slices, making it near-impossible to run rail analysis on all slices due to long runtimes. Using MAVIREC as a fast IR drop estimator is not a viable solution either as each inference involves extracting temporal features at recurring runtime costs for all slices. This calls for techniques to select a subset (70–200) of slices that represent the design and worst-case IR drop

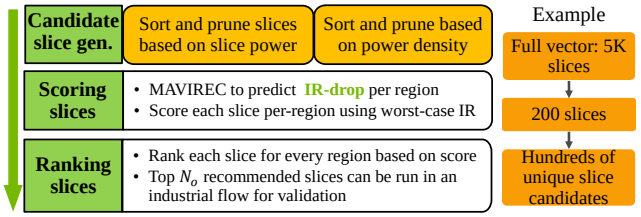

Fig. 3: MAVIREC vector profiling which uses ML inference to generate hundreds of ranked candidates.

vector. In contrast, industrial flows are limited to 3–5 slices due to IR analysis runtimes.

Fig. 3 and Algorithm 1 brief MAVIREC's vector profiling flow. Inspired by classical ML recommender system [7] nomenclature, our method (Fig. 3) consists of (i) candidate generation (two stages), (ii) scoring, and (iii) ranking. The two stages of candidate generation prune out slices based on average power and regional-power respectively. The remaining slices are scored based on worst-case IR drop from ML inference and ranked to maximize regional coverage.

Algorithm 1 MAVIREC's algorithm for vector profiling

Input: Per-instance toggle count T_c; p_i, p_s, p_l; Number slices to select N_a, N_r, N_o; Region size $w \times l$; Chip dimensions $W_c \times L_c$; List of candidate slices in vector C_N;
Output: Slices with worst IR drop C_{N_o}; IR_{chip,N_o}

1: $W_r = \frac{W_c}{w}$ and $L_r = \frac{L_c}{l}$
2: *//Candidate slice generation stage 1: sort and eliminate using average slice power*
3: **for** each slice c in C_N **do**
4: $\quad P_{slice}[c] = \sum_{\forall \text{instances}} \left[p_l + \frac{T_c[c]}{20} (p_s + p_i) \right]$
5: **end for**
6: C_{N_a} = Top N_a candidate slices with highest value in P_{slice}
7: *// Candidate slice generation stage 2: sort and eliminate using average region power*
8: **for** each region r_i in $W_r \times L_r$ **do**
9: $\quad$ **for** each candidate slice c in C_{N_a} **do**
10: $\quad\quad P_R[r_i][c] = \sum_{\forall \text{instances} \in r_i} \left[p_l + \frac{T_c[c]}{20} (p_s + p_i) \right]$
11: $\quad$ **end for**
12: $\quad C_{N_r}[r_i]$ = Top N_r candidate slices with highest value in $P_R[r_i]$
13: **end for**
14: C_{N_c} = Unique candidates from C_{N_r}
15: *// Candidate scoring and ranking starts here*
16: **for** each slice c in C_{N_c} **do**
17: $\quad F[c]$ = Feature_extraction(c); IR_{chip} = ML_inference($F[c]$)
18: $\quad$ **for** each region r_i in $W_r \times L_r$ **do**
19: $\quad\quad IR_{score}[r_i][c]$ = max($IR_{chip}[c]$) in r_i
20: $\quad$ **end for**
21: **end for**
22: $n = 0$
23: **while** $n \leq N_o$ **do**
24: $\quad r_i, C_{N_o}[n]$ = Top N_o unique candidates, regions with highest IR_{score} values
25: $\quad IR_{chip,N_o}[n] = IR_{score}[x][C_{N_o}[n]] \; \forall \; x \; \epsilon \; W_r \times L_r$
26: $\quad n$++
27: **end while**

Candidate generation: This step consists of two stages (Lines 3–14).

First stage In each vector, we observe thousands of slices with near-zero switching activity. Therefore, we prune the slices based on their average power (lines 3–6) eliminating obviously non IR-critical slices. The pruned and sorted list, C_{N_a}, has hundreds ($>$ 3–5 in industrial flows) of candidates for the next stage.

Second stage Since the IR drop of an instance depends on its switching activity and neighborhood, power per region is vital. Therefore, we calculate the power per region (lines 8–14) and rank the C_{N_a} candidates in each region. Next, we extract the top N_r from each region resulting in a list, C_{N_r}, of $N_r \times W_r \times L_r$ candidate slices and C_{N_c} is a list of N_c unique candidates from C_{N_r}.

Candidate slice scoring and ranking For each candidate in C_{N_c} we generate all the features (Table I) and use MAVIREC ML inference to generate a full-chip IR drop (line 17). The worst-case IR drop across the chip is used to score each slice. This results in $N_c \times W_r \times L_r$ score values in IR_{score}(line 19). We then rank each candidate based on the score and record its corresponding region (line 24). Only those unique N_o candidates that correspond to the worst-case IR drop of a previously uncovered region are stored in C_{N_o}.

TABLE II: Summary of designs and vectors used in MAVIREC experiments with a tile size of 2.5μm $\times$ 2.5μm.

Design	#inst. (mill.)	T1		T2		T3	
		%IR -critical regions	Toggle rate	%IR -critical regions	Toggle rate	%IR -critical regions	Toggle rate
D1	3.26	13.66	0.054	16.04	0.062	8.98	0.041
D2	2.19	4.60	0.043	4.24	0.040	3.59	0.038
D3	2.18	4.28	0.040	4.31	0.038	3.88	0.038
D4	2.43	16.88	0.085	10.68	0.087	13.00	0.089

TABLE III: Performance of MAVIREC as an IR drop predictor. RMSE and MAE at instance-level granularity and % accuracy as a binary classifier at region-based granularity when compared to an industrial flow.

		RMSE, MAE (mV)	% Accuracy 1x1, 6x6			RMSE,MAE (mV)	% Accuracy 1x1, 6x6
D1	**T1**	4.44, 33	89.09, 84.98	**D3**	**T1**	3.02, 26.8	96.26, 93.45
	T2	4.61, 25.2	85.49, 88.72		**T2**	2.99, 22.4	96.12, 92.89
	T3	3.91, 30.2	92.93, 88.93		**T3**	2.93, 35.1	95.96, 92.56
D2	**T1**	3.1, 25.4	95.68, 92.73	**D4**	**T1**	3.87, 24.1	89.43, 91.07
	T2	3.11, 20.6	95.86, 92.77		**T2**	3.38, 24.7	90.27, 93.62
	T3	2.99, 21.8	96.08, 92.19		**T3**	3.31, 24.5	90.25, 94.55

For each region, the slices are ranked in decreasing order of the score in the region. The unique slice from each region with the highest score is selected such that regional coverage is maximized. We also report the IR drop maps, IR_{chip}, of C_{N_o} candidate list (line 25).

IV. Results and Discussions

In our experiments, we use four industrial designs, D1–D4, implemented in a sub-10nm FinFET technology and three multi-cycle vectors, T1–T3, per design summarized in Table II. The data available to us were taped out designs with low IR drop values, requiring use of an 8mV threshold to classify IR-critical regions. The model is trained using golden per-instance IR drop labels obtained from an industrial flow using commercial tools. We evaluate MAVIREC with *leave one out cross validation*. MAVIREC is implemented within a PyTorch 1.6 framework on an 8-core CPU machine with 256GB RAM and one NVIDIA Tesla V100GPU with 32GB RAM. An ADAM optimizer with an L_2 regularizer is used for training. The model takes 9 hours to train and is a one-time cost per technology as it is transferable across both designs and vectors.

A. MAVIREC for IR drop prediction and classification

MAVIREC vs. industrial flow for IR drop We compare MAVIREC-predicted IR drop against ground truth IR drop from an industrial flow. Like an industrial flow, MAVIREC predicts an IR drop per-instance. Table III shows the RMSE and max error for all the designs and tests. The RMSE is very small ($<$4mV) and within reasonable limits for instance-level applications such as cell-level IR drop mitigation and IR-aware STA. The table also depicts MAVIREC's performance as an IR drop hotspot classifier on a per-tile basis, i.e., if the average IR drop of all instances in that tile is greater than the threshold. We consider two granularities, a 1×1 tile (2.5μm $\times$ 2.5μm) and 6×6 tiles (15μm $\times$ 15μm), and report the accuracy. At 6×6 granularity we have an F1 score of 0.78, despite the heavily imbalanced dataset ($<$10%). This accuracy outperforms prior art and is sufficient for vector profiling (Section IV).

For a visual comparison, we convert the instance-based $\mathbf{IR_i}$ to tile-based. Fig. 4 shows the predicted and ground truth IR maps for a section of each design. The MAVIREC-predicted IR drop map captures all major hotspots relative to the ground truth.

MAVIREC vs. ML-based IR drop classifiers As a comparison to prior art, we implemented a maximum CNN structure similar to PowerNet [3]. To ensure a fair comparison, we make the following changes: (i) As the focus of PowerNet was vectorless IR drop estimation and does not support the use of multi-cycle switching activity, to adapt PowerNet, we take the average toggle rate from the vectors. (ii) PowerNet predicts IR drop on a per-tile (1μm $\times$

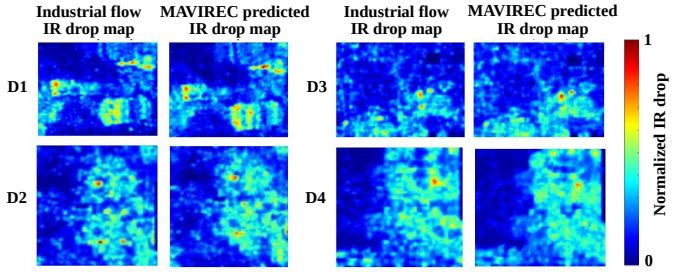

Fig. 4: MAVIREC-predicted IR maps vs industrial flow IR maps.

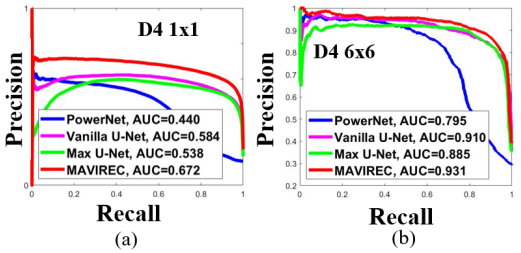

Fig. 5: PR curves comparing MAVIREC against other ML models in predicting the minority class for tile sizes of (a) 1x1 and (b) 6x6 tiles.

1μm) basis, while MAVIREC is on a per-instance basis. Therefore, MAVIREC generates a region-based IR drop, like PowerNet, by taking the mean of the predicted-IR drop of all instances in the tile.

We compare MAVIREC against two other ML models: (i) a vanilla U-Net, identical to MAVIREC except that it uses 2D convolutional layers, and (ii) a max U-Net, identical to PowerNet except that the CNN is replaced with a 2D convolutional U-Net with an output regression layer (Section II). The vanilla U-Net processes all time steps simultaneously while the max U-Net processes each time step separately. The max U-Net sets the final IR drop to the maximum IR drop for each instance across all time-steps.

A high area under curve (AUC) of the precision-recall (PR) curve demonstrates the ability of the classifier to predict the minority class, in a highly imbalanced dataset. Fig. 5 shows that MAVIREC outperforms both PowerNet and other ML models in predicting the IR-critical class for 1×1 (Fig. 5(a)) and 6×6 (Fig. 5(b)) tile granularities. The 3D convolutional layers in MAVIREC capture the sparse switching activity, but the vanilla and Max U-Net fail to do so due to their 2D convolutional nature.

TABLE IV: Runtime comparison of industrial flow and models.

Task	Industrial flow	PowerNet	Max U-Net	Vanilla U-Net	MAVIREC
Feature extraction	3 hours	17 mins			
ML inference		5 mins	7.2s	1s	3s

Table IV compares runtimes of MAVIREC vs. an industrial flow, PowerNet, MAVIREC, and other ML models on D1. With a runtime of 18 minutes, MAVIREC has a **10× speedup** over a standard rail analysis. Feature extraction dominates the runtime with MAVIREC ML inference $< 1\%$. MAVIREC is $2\times$ faster than max-U-Net and **100× faster** than PowerNet in pure inference. The vanilla U-Net is faster than MAVIREC but is sub-optimal (Fig. 5).

B. MAVIREC for vector profiling

Quality of MAVIREC-recommended vectors The input vectors to our profiling algorithm (Section III) have about 100,000 clock cycles $\approx$ 5000 slices. For $N_a = 200$, $N_r = 5$, $N_o = 3$, and $w \times l = 15\mu\text{m} \times 15\mu\text{m}$ (6×6 tiles), the algorithm generates $N_c \approx 100$ candidate slices per vector. Fig. 6 shows the coverage comparison between the industrial flow-generated top-3 candidate slices and 168 MAVIREC-recommended slices on D2. Fig. 6 (leftmost) compares the regional coverage. The middle picture shows the part of the design with the largest cluster of industrial flow-missed IR-critical spots

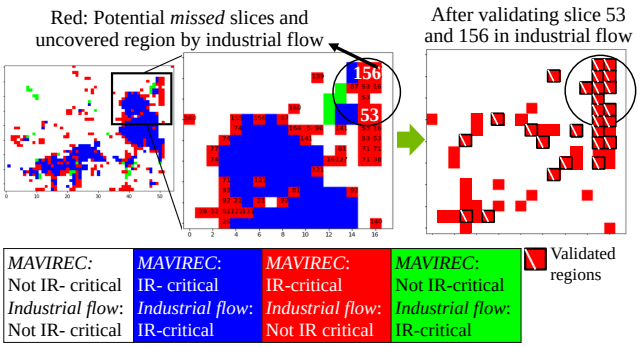

Fig. 6: Regional coverage comparison of MAVIREC's 168 candidate slices against 3 slices from industrial flow on design D2.

TABLE V: MAVIREC vs. industrial flow recommended slices.

Design, Test	Industrial flow profiling		MAVIREC profiling		Comparison			
	N_c	#IR-critical regions	N_c	#IR-critical regions	#Regions IR-critical in both	%Regions uncovered MAVIREC	%Regions uncovered industrial flow	#Unique slice IDs of missed regions
D1, T1	3	1163	133	1531	1093	1.7	10.8	30
D2, T1	3	422	168	550	383	1.6	6.9	36
D3, T1	3	369	166	553	329	1.7	9.4	26
D4, T1	3	682	73	786	672	0.4	4.5	11

(set of red regions). The numbers on the red regions indicate slice IDs (a numeric ID number for identifying each unique slice of the testbench) that resulted in those regions being reported as IR critical by MAVIREC. These are a set of *missed* slices and we validate slice IDs 53 and 156 using an industrial flow. The rightmost map shows black-outlined and white-dashed red regions which were validated to be IR-critical. We limit our validation to missed slices with the largest uncovered region cluster given large industrial flow runtimes.

Table V lists the number of candidates, N_c, generated for each design and T1. For each with a region size of 6×6 tiles, we have 70–170 candidate slices generated while a industrial flow generated three slices. MAVIREC provides a large coverage by reporting an average of $\sim 5\%$ of the regions as potentially uncovered by the industrial flow and has less than 1.7% false negatives.

MAVIREC vector profiling runtimes MAVIREC is able to provide high-quality recommendations for a 100K-cycle vector in 30 minutes which corresponds to a $4\times$ speedup vs. the industrial flow.

V. Conclusion

We propose MAVIREC: a vector profiling system that provides recommendations for worst-case IR drop switching patterns using an ML model for IR drop prediction. While this work focused on vectored dynamic IR drop analysis, the ML inference engine can be adopted for both vectorless dynamic and static IR drop analysis.

References

[1] N. Ahmed, M. Tehranipoor, and V. Jayaram, "Transition delay fault test pattern generation considering supply voltage noise in a SOC design," in *Proc. DAC*, 2007.

[2] Y.-C. Fang *et al.*, "Machine-learning-based dynamic IR drop prediction for ECO," in *Proc. ICCAD*, 2018.

[3] Z. Xie *et al.*, "PowerNet: Transferable dynamic IR drop estimation via maximum convolutional neural network," in *Proc. ASP-DAC*, 2020.

[4] V. A. Chhabria *et al.*, "Thermal and IR drop analysis using convolutional encoder-decoder networks," in *Proc. ASP-DAC*, 2021.

[5] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assisted Intervention*, 2015.

[6] V. A. Chhabria *et al.*, "MAVIREC: ML-aided vectored IR-drop estimation and classification," *arXiv:2012.10597 [cs.ar]*, 2020.

[7] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for YouTube recommendations," in *Proc. ACM Conf. Recomm. Sys.*, 2016.