Оценка карты разводимости при проектировании цифровых блоков СБИС с помощью графовых нейронных сетей

Н. В. Желудков[1], Я. М. Карандашев[2], Е. С. Кочева[1], М. Х. Сайбодалов[1], З. Б. Сохова[1], А. А. Умнова[1]

[1]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, nvgel@cs.niisi.ras.ru;

[2] ФГУ ФНЦ НИИСИ РАН, Москва, Россия, karandashev@niisi.ras.ru

Аннотация. В рамках данной работы рассматривается решение задачи оценки карты разводимости на ранних этапах топологического проектирования цифровых блоков СБИС с помощью применения нейросетевой модели машинного обучения, основанной на графовой нейронной сети. Раннее предсказание проблемных мест с разводкой позволит разработчику топологии изменить такие характеристики проектируемого блока, как план размещения, расположение макроблоков, а также входных и выходных портов таким образом, чтобы предотвратить возникновение проблем с трассировкой соединений на поздних этапах, тем самым сократив число запусков САПР и общее время проектирования схемы. Применение графовых нейронных сетей позволяет учитывать дополнительную информацию о связях элементов в нетлисте, для более точного предсказания.

Ключевые слова: СБИС, топологическое проектирование, карта разводимости, машинное обучение, графовые нейронные сети

1. Введение

Одним из этапов топологического проектирования цифровых блоков СБИС является оценка разводимости межсоединений элементов на кристалле. Данный этап в САПР заключается в следующем: происходит разделение площади схемы на равные участки в виде квадратов со стороной в несколько высот стандартных ячеек, оценивается число межсоединений, которые должны проходить через этот квадрат, определяется число доступных для разводки треков металлизации в этой области, вычисляется отношение первого значения ко второму. Если это соотношение не превышает 100%, то нарушение разводимости в рассматриваемой области отсутствует – доступного числа треков хватает для разводки требуемого числа трасс металлизации. При значении этого соотношения больше 100% существенно вырастает вероятность получить нарушение DRC-правил (правил проектирования для выбранной технологии) после проведения этапа детальной трассировки. В рамках данной работы мы будем называть данное соотношение как “разводимость”, а агрегацию локальных значений этого параметра по всей площади – “картой разводимости”. Аналогичным параметром в англоязычной литературе и САПР является “congestion”.

Оценка разводимости в современных САПР проводится после прохождения двух этапов топологического проектирования:

- размещения стандартных ячеек (включая глобальное и детальное размещение);
- глобальной трассировки, которая заключается в оценочной трассировке межсоединений без сильной привязки к трекам.

Таким образом, получить карту разводимости, чтобы оценить проблемные места, можно только после прохождения в САПР перечисленных выше этапов, что может занимать для современных блоков СБИС с сотнями тысяч и миллионов логических вентилей значительное время. Это создает следующую проблему: для оценки разводимости при любом изменении плана размещения и новой расстановкой макроблоков, при изменении в сетке земли-питания, а также изменении изначального нетлиста схемы – требуется запускать новую итерацию маршрута в САПР с прохождением этапов размещения стандартных ячеек и глобальной трассировкой. Запуск каждой итерации САПР для оценки карты разводимости на новом плане размещения увеличивает общее время проектирования схемы.

Основной задачей, решаемой в рамках данной работы, является создание модели машинного обучения для предсказания карты разводимости без прохождения времязатратных этапов размещения стандартных ячеек и глобальной трассировки. Входными данным для этой мо-

дели являются изначальный нетлист проектируемой схемы, а также характеристики стандартных ячеек, представленных в технологической библиотеке – число пинов и площадь ячеек. Для получения выходной метрики (значения разводимости на локальных участках) для обучения модели был создан маршрут топологического проектирования в САПР с открытым исходным кодом OpenROAD [1], включающий в себя прохождение этапов размещения ячеек, глобальной трассировки и выписки реальной карты разводимости. На Рисунке 1 представлено изображение карты разводимости, полученной в САПР OpenROAD.

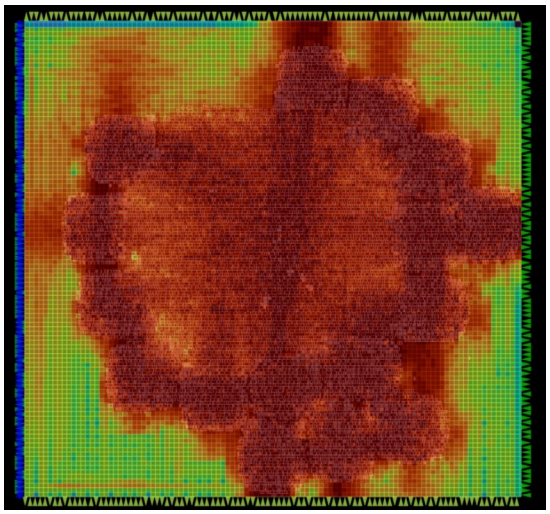

Рис. 1. Карта разводимости в OpenROAD

Более темным участкам соответствуют места на плане размещения с большим значением разводимости.

Основными причинами нарушения разводимости можно считать:

- присутствие в нетлисте логических вентилей с большим числом пинов, что может сказаться на разводимости в том месте, где будут размещены в дальнейшем эти ячейки, особенно если несколько таких ячеек будут расположены рядом друг с другом;
- высокая плотность сетки земли-питания, трассы которой занимают большое число треков, доступных для разводки;
- размещение большого числа стандартных ячеек в узких каналах между макроблоками и рядом с их краями;
- плотное размещение входных и выходных портов, что влечет за собой повышенную плотность разводки в этой области.

Для обучения и тестирования модели был выбран набор из 25 блоков, находящийся в открытом доступе (openABC) [2] и представленный в виде RTL описания. Был произведен логический синтез блоков в САПР Genus на основе технологии “Микрон 180 нм”, а также выполнено топологическое проектирование по этим же проектным нормам для получения реальных значений карт разводимости.

В своей работе мы опирались на статьи Кирби и др. [3] и Гозе и др. [4]. В этих работах авторы получали данные из нетлистов и представляли их в качестве графов, где у каждого узла есть некоторые атрибуты. В данной работе используется тот же подход: из нетлистов мы получали данные о площади ячеек и количестве пинов для каждой ячейки и использовали эти данные для обучения наших моделей. Значения разводимости, полученные из карты разводимости, использовались в качестве меток. Именно эти значения мы будем предсказывать с помощью нейронных сетей в п.3.

2. Методы

2.1. Графовые нейронные сети

Графовые нейронные сети (GNN) представляют собой класс нейронных сетей, предназначенных для обработки данных, представленных в виде графа. Основная идея GNN заключается в обновлении признаков каждой вершины графа на основе признаков ее соседей. GCN (Graph Convolutional Network) является самым распространенным видом графовых нейронных сетей. На каждом слое GCN вершины обмениваются информацией с их соседями и обновляют свои признаки. В данной работе мы применяли два типа графовых слоев: GATConv [5] и SAGEConv [6] из фреймворка PyTorch Geometric [7].

Рассмотрим граф $G = (V, E)$, где G – набор узлов, а E – набор ребер. Матрицей смежности A мы называем матрицу, в которой $A_{ij} = 1$, если существует ребро $(i, j) \in E$.

Базовый слой SAGEConv определяется следующим образом:

$$h_i^{(k)} = f^{(k)}(W_1^{(k)} h_i^{(k-1)} + W_2^{(k)} MEAN\left(h_j^{(k-1)}\right), \quad (1)$$

где $W^{(k)}$ – матрица весов, а $f^{(k)}$ – нелинейная функция активации, например, ReLU.

Слой GATConv, в свою очередь, реализует концепцию внимания (attention), аналогично тому, как этот механизм используется в других классах нейронных сетей:

$$e_{ij}^{(k)} = \text{LeakyReLU}\left(\left[W^{(k)} h_i^{(k-1)} || W^{(k)} h_j^{(k-1)}\right]^T a^{(k)}\right), \quad (2)$$

$$\alpha_{ij}^{(k)} = \mathrm{softmax}_j\left(e_{ij}^{(k)}\right) = \frac{\exp\left(e_{ij}^{(k)}\right)}{\sum_{r\in\mathcal{N}_i}\exp\left(e_{ir}^{(k)}\right)}, \quad (3)$$

$$h_v^{(k)} = f^{(k)}\left(W^{(k)}\sum_{j\in\mathcal{N}_i}\alpha_{ij}^{(k)}\,h_j^{(k-1)}\right). \quad (4)$$

2.2. Обработка данных

Каждый элемент, полученной ранее схемы, содержащийся в нетлисте, рассматривается как узел в графе. Рёбра представляют собой взаимосвязи между элементами схемы, как определено в проводах нетлиста. Мы предполагаем, что набор узлов в каждом проводе полносвязный. Входные и выходные порты удаляются из графа, так как они обычно размещаются вручную, и их высокие степени узлов по сравнению со стандартными ячейками негативно сказываются на эффективности обучения. Таким образом, в качестве элементов схемы мы оставляем только ячейки. Провода со степенью более 10 исключаются из итогового графа, так как они вводят клики, слишком большие для эффективной работы с ними. Полученный нетлист является графом, так как он представляет из себя набор ячеек и соединения между ними. Соответственно, мы можем представить нетлист как граф $G = (V, E)$, где V – набор узлов, представляющий ячейки и E – набор ребер. Каждый узел V имеет набор атрибутов $X \in R^3$, состоящий из значений площади ячейки (area) и количества соединений (num_pins) – входные параметры, а также значения разводимости (congestion value) – значения, которые мы собираемся предсказывать с помощью нейронных сетей. Значение разводимости мы получаем из карты разводимости путем присвоения ячейкам значений из соответствующих квадратов на карте, так как каждая ячейка попадает в некоторый квадрат на карте разводимости. Таким образом, были обработаны все 25 нетлистов и получено столько же графов для дальнейшего обучения моделей. Характеристики получившихся графов можно увидеть в Приложении 1. Хотя показатель разводимости в большинстве случаев варьируется в пределах от 0 до 100, обнаружен граф, где этот показатель достигает 518. Данный граф был исключен из выборки, так как подобные значения негативно влияют на обучение модели.

3. Результаты

В качестве тестовой выборки были выбраны 5 графов. Мы сравнили результаты моделей, в которых использовали слои GAT, SAGE и простой линейный персептрон с методом аппроксимации Neighbourhood size, в котором мы вычисляем количество соседей в окрестности 5 переходов, как было предложено в [8]. Результаты, усредненные по 10 обучениям моделей, представлены в Таблице 1. В качестве метрик работы алгоритмов мы приводим значения корреляции Кендалла и средней абсолютной ошибки (MAE).

Корреляция Кендалла вычисляется путем оценки того, является ли порядок выборки пар одинаковым как в реальных, так и в прогнозируемых значениях для всех возможных пар. То есть это разница между процентом совпавших и инверсных пар. Таким образом, необязательно, чтобы распределение предсказаний соответствовало распределению реальных значений.

Средняя абсолютная ошибка – это среднее абсолютных разностей между целевым значением и значением, предсказанным моделью. Таким образом, мы видим, насколько в среднем модель ошибается в каждом предсказании. На основании трех обученных алгоритмов были построены карты предсказаний разводимости. Результаты представлены на Рисунке 2.

Таблица 1. Результаты работы моделей

	ac97_top	aes128_core	dft_top	eth_top	fpu
	Корреляция Кендалла				
GAT(3 слоя, 2 MLP)	0.137 +- 0.005	0.239 +- 0.025	0.250 +- 0.014	0.076 +- 0.013	0.202 +- 0.009
SAGE(3 слоя, 2 MLP)	0,155 +- 0.007	0,294 +- 0.007	0.280 +- 0.008	0.048 +- 0.029	0.184 +- 0.010
Linear(2,1)	0.140 +- 0.022	0.207 +- 0.571	0.144 +- 0.196	0.053 +- 0.381	0.169 +- 0.644
Neighbourhood metric	0.198	-0.052	0.482	0.232	0.194
	Средняя абсолютная ошибка				

GAT(3 слоя, 2 MLP)	13.52 +- 0.33	11.49 +- 0.57	10.86 +- 0.19	10.96 +- 0.38	10.35 +- 0.64
SAGE(3 слоя, 2 MLP)	12.73 +- 1.01	9.83 +- 0.64	10.70 +- 0.70	11.73 +- 1.01	8.10 +- 0.15
Linear(2,1)	16.26 +- 3.21	9.29 +- 2.85	12.02 +- 0.50	10.87 +- 1.18	17.83 +- 3.82

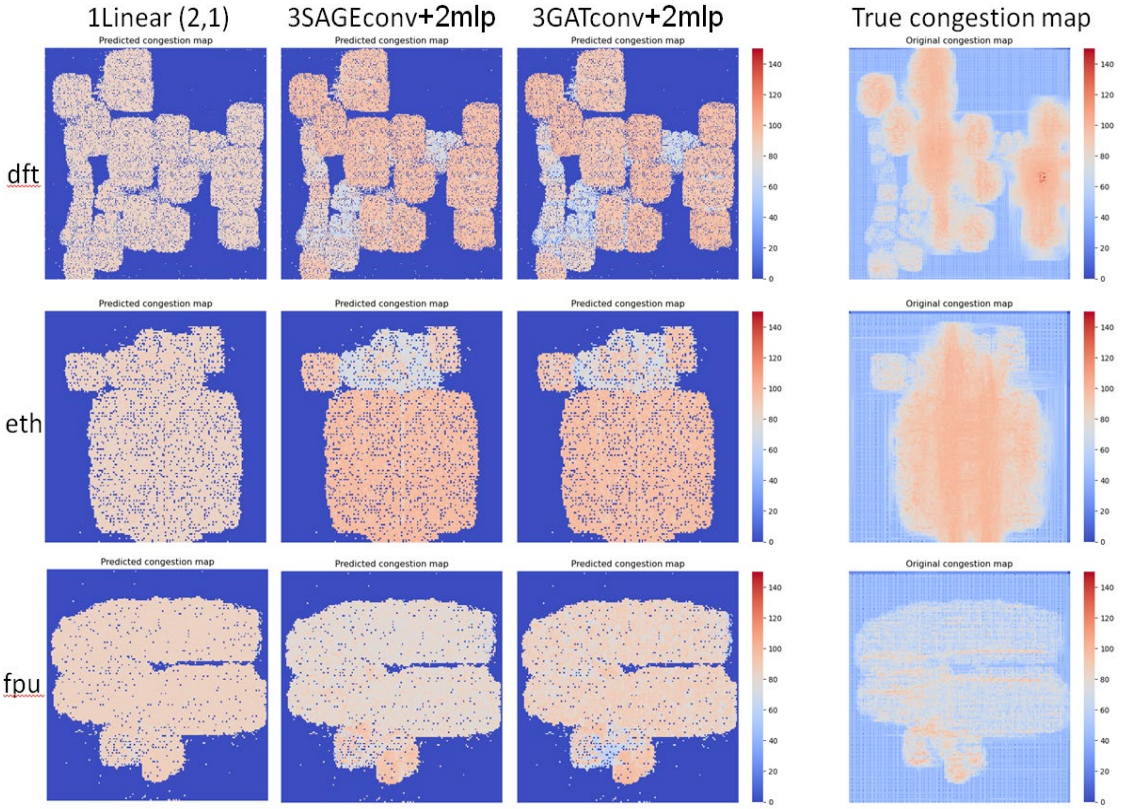

Рис. 2. Карты предсказаний, построенные на основании предсказаний моделей

4. Заключение

В ходе текущего исследования было обработано 25 графов нетлистов, общее количество вершин в которых достигает приблизительно 382 тысячи, используется 753 библиотечных элемента (DEF). Одно из наблюдений заключается в том, что даже без применения обучения, простое предсказание на основе среднего значения может обеспечить абсолютную ошибку в пределах от 10 до 18 по всем графам. Основной проблемой является выбор способа разделения данных на обучающую и тестовую выборки. Некоторые графы значительно различаются как по значению разводимости, так и по числу вершин и используемым элементам. Графы, которые сильно отличаются, могут негативно влиять на результаты тестирования. Отказ от таких графов в пользу более схожих может значительно улучшить результаты. Сравнение с исследованиями Кирби [3] и Гозе [4] и др. выявило потенциальные направления для дальнейшего развития: во-первых, возможность увеличения обучающей выборки данных в 100 раз и, во-вторых, интеграция входных эмбеддингов, основанных на матричном разложении, или обучаемых эмбеддингов для различных типов ячеек.

Публикация выполнена в рамках государственного задания ФГУ ФНЦ НИИСИ РАН по теме № FNEF-2022-0008.

Estimation of Congestion Map in the VLSI Design of Digital Blocks with Graph Neural Network

N. V. Zheludkov, I. M. Karandashev, E. S. Kocheva, M. K. Saibodalov, Z. B. Sokhova, A. A. Umnova

Abstract. This paper considers a solution to the problem of estimating the congestion map in the early stages of VLSI layout design of digital blocks by applying a neural network model of machine learning based on a graph neural network. Early prediction of congestion problems will allow the layout engineer to modify design block characteristics such as floorplan, IP-block`s placement and input-output ports to prevent interconnect routing issues at later stages, thereby reducing the number of CAD runs and overall circuit design runtime. The application of graph neural networks allows to take into account additional information about the connections of elements in the netlist for more accurate prediction.

Keywords: VLSI, layout design, congestion map, machine learning, graph neural networks

Литература

1. [Электронный ресурс]. URL: https://theopenroadproject.org/ (дата обращения: 10.11.2023)

2. A.B. Chowdhury, B. Tan, R. Karri, S. Garg. OpenABC-D: A Large-Scale Dataset for Machine Learning Guided Integrated Circuit Synthesis. (2021). ArXiv preprint (2021) arXiv:2110.11292.

3. R. Kirby, S. Godil, R. Roy, B. Catanzaro. Congestionnet: Routing congestion prediction using deep graph neural networks. «2019 IFIP/IEEE 27th International Conference on Very Large Scale Integration (VLSI-SoC)», 2019, pp. 217–222.

4. A. Ghose, V. Zhang, Y. Zhang, D. Li, W. Liu, M. Coates. Generalizable cross-graph embedding for GNN-based congestion prediction. «2021 IEEE/ACM International Conference on Computer Aided Design (ICCAD)», 2021, pp. 1–9.

5. P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio. Graph attention networks. ArXiv preprint (2017) arXiv:1710.10903.

6. W.L. Hamilton, Z. Ying, J. Leskovec. Inductive representation learning on large graphs. «Neural Information Processing Systems», 2017.

7. M. Fey, J.E. Lenssen. Fast Graph Representation Learning with PyTorch Geometric. «ICLR Workshop on Representation Learning on Graphs and Manifolds», 2019.

8. P. Kudva, A. Sullivan, W. Dougherty. Metrics for structural logic synthesis. «IEEE/ACM International Conference on Computer Aided Design», 2002, pp. 551−556.

Приложение 1. Характеристики нетлистов собранного датасета (белым цветом отмечены графы из обучающей выборки, серым из тестовой)

netlist_name				**AREA**			**NUM_PINS**			**CONGESTION_VALUE**		
	кол-во узлов	**кол-во рёбер**	**библиотечных элементов**	**min**	**max**	**mean**	**min**	**max**	**mean**	**min**	**max**	**mean**
aes_cipher_top	8 419	104366	112	8.19	65.54	26.07	2	9	4.23	18.18	100.00	90.89
des3	1 432	20550	123	8.19	65.54	26.82	2	9	4.54	46.67	100.00	90.83
dynamic_node_top	5 757	79219	129	8.19	73.73	37.88	2	9	4.70	54.55	136.36	91.94
FIR_filter	951	2838	22	8.19	69.63	58.08	2	6	4.40	13.64	86.36	56.19
i2c_master_top	416	3505	71	8.19	65.54	34.25	2	7	4.27	23.33	95.45	63.75
i_buf	66 279	592928	353	8.19	159.74	26.44	2	9	3.64	22.73	204.55	91.72
idft_top	73 584	984272	162	8.19	114.69	42.82	2	9	4.66	26.67	154.55	82.57
IIR_filter	1 524	4474	27	8.19	69.63	55.31	2	9	4.27	22.73	90.91	56.89
mc_top	2 974	30178	165	8.19	77.82	35.84	2	9	4.34	0.00	100.00	73.09
pci_bridge32	6 572	75934	201	8.19	77.82	41.28	2	9	4.63	27.27	100.00	74.49
pcm_slv_top	173	1473	21	8.19	73.73	39.87	2	7	4.21	27.27	77.27	56.60
sasc_top	256	1786	49	8.19	77.82	37.66	2	7	4.29	26.67	81.82	58.97
sha256	4 369	51316	94	8.19	69.63	37.26	2	9	4.33	53.33	106.67	92.44
simple_spi_top	340	2621	66	8.19	65.54	35.13	2	7	4.23	27.27	86.36	63.07
spi_top	1 261	13255	86	8.19	163.84	28.99	2	12	4.25	31.82	100.00	77.88
tv80s	2 613	29844	143	8.19	69.63	28.38	2	9	4.26	26.67	100.00	80.91
usb_phy	269	1732	60	8.19	98.30	32.11	2	7	3.77	36.36	86.67	59.24
~~vga_enh_top~~	~~33 931~~	~~472958~~	~~198~~	~~8.19~~	~~69.63~~	~~40.16~~	~~2~~	~~9~~	~~4.65~~	~~22.73~~	~~518.18~~	~~140.54~~
wb_conmax_top	15 137	248335	84	8.19	73.73	24.31	2	9	4.47	63.33	181.82	124.43
wb_dma_top	1 531	15291	93	8.19	65.54	34.50	2	9	4.21	0.00	100.00	82.80
ac97_top	4 417	51376	100	8.19	65.54	39.78	2	9	4.44	27.27	100.00	70.43
aes128_core	15 609	158622	170	8.19	65.54	26.71	2	9	4.14	59.09	133.33	95.79
dft_top	73 651	981628	166	8.19	114.69	42.73	2	9	4.65	22.73	172.73	82.24
eth_top	21 377	270700	229	8.19	77.82	40.35	2	9	4.68	33.33	104.55	88.25
fpu	39 541	199528	569	8.19	139.26	24.50	2	9	3.06	26.67	100.00	67.97
ВСЕГО:	**382 383**	**4 398 729**	**753**	**8.19**	**163.84**	**35.89**	**2**	**12.00**	**4.29**	**0.00**	**518.18**	**80.56**