ISSN 0002-306X. Proc. of the RA NAS and NPUA Ser. of tech. sc. 2024. V. LXXVII, N3

UDC 621.3.049.77 MICROELECTRONICS

DOI: 10.53297/0002306X-2024.v77.3-331

A.A. GHAZARYAN

THE SKEW MINIMIZATION METHOD USING DIFFUSION MODELS

Integrated circuit (IC) functionality increases significantly, which force to have more strict requirements, especially in fast systems. In digital methodology the clock tree synthesis has crucial importance, which affects the IC's performance. One of the clock tree parameters is clock skew, which should be minimum, otherwise it will affect the IC's performance. Minimal skew development is the tradeoff between timing margins, routing congestion, etc. Currently, there are different methodologies integrated in physical design tools, which are aimed at parameters to find good trade-off points. Some of the algorithms do not work properly in current processes, and human should act manually, which makes electronic design automated tools work not automatically. Fix such issues, a skew minimization method with the use of diffusion models is presented. With the proposed method, the runtime has increased by 5% in respond to EDA tools integrated methodologies, and decreased in terms of AI tools, and skew decreased by 28%.

Keywords. clock tree synthesis, skew, routing congestion, machine learning, diffusion models.

Introduction. Clock tree synthesis (CTS) is a crucial component in IC design that guarantees the performance and functionality of digital integrated circuits. As seen in Fig. 1, the clock signal, also referred to as a chip's "heartbeat," synchronizes all the design's operations.

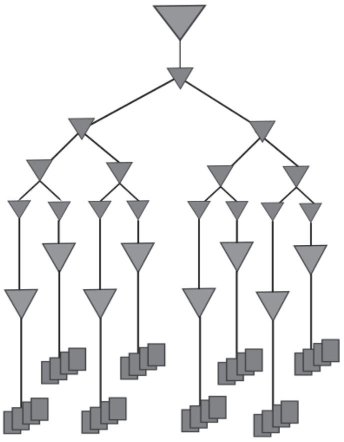


Fig. 1. The general view of a clock tree

One of the main objectives of clock tree synthesis is skew minimization, which minimizes the variation in the arrival times of the clock signal at different components. A clock skew example for two registers is shown in Fig. 2. The skew is critical for maintaining synchronization and optimal performance in synchronous digital circuits [1,2].

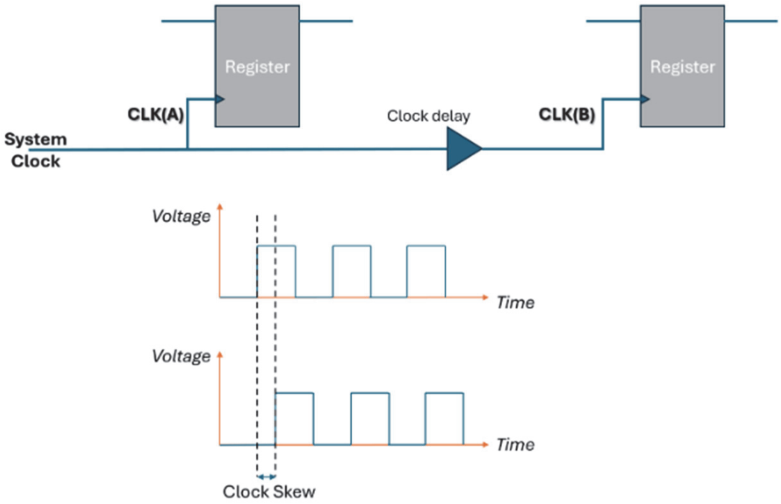


Fig. 2. A clock skew

Even a small clock skew can cause serious timing problems in high-frequency modern systems, such as setup and hold times. Therefore, attaining a well-balanced clock tree is crucial to meet timing standards, which will become more stringent as technology advances. CTS also has difficulties in controlling power usage, routing congestion, and clock latency. Because clock networks frequently consume a significant amount of integrated circuit power, performance and energy efficiency must be carefully optimized and balanced [3].

To overcome these obstacles, the clock tree must be efficiently optimized using techniques and tools that ensure all clocked elements receive the clock signal at the same time, preserving synchronization and improving the system performance. But there are trade-offs associated with each strategy that need to be considered. The most widely used approaches, which can be found in the references are listed below:

- Buffered clock tree synthesis.

According to the method, buffers (used inverter pairs) are used to balance the delays and reduce the skew. By placing buffers, it is possible to achieve zero skews while minimizing the total wire length, which improves timing performance. The main picture of the buffered clock tree synthesis is shown in Fig 3.

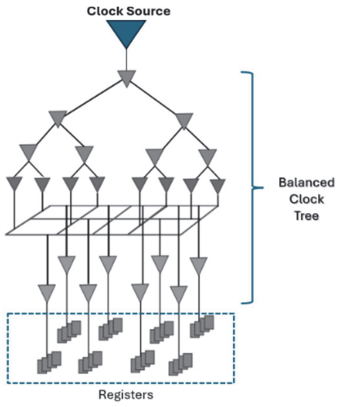


Fig. 3. The balanced clock tree example

The buffers increase the overall power consumption of the clock network. Additionally, the buffer insertion process can introduce routing congestion, especially in dense designs [4].

• Buffer Resizing and relocation.

The method adjusts buffer sizes and positions to optimize the clock arrival times, reduces the insertion delay and minimizes the skew. It enhances the timing performance of design by improving the setup and hold time margins. An example of design before and after optimization is shown in Fig. 4.

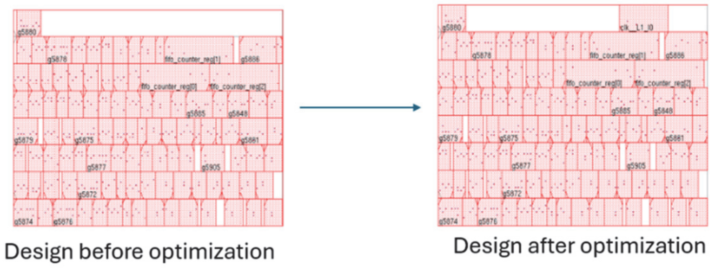


Fig. 4. Design before and after optimization

The method requires more resources in terms of routing and wire length compared to pure tree-based structures. The corresponding will be a reason for overlaps, as a result, the re-placement will be needed, in Fig 4, notably, replacement after CTS optimization has been done. And, achieving the optimal buffer configuration can be computationally expensive, especially for large designs [5].

• Mixed tree-mesh clock distribution networks

Fig. 5 illustrates an overview of those networks, which combine the benefits of mesh and tree topologies to reduce the local clock skew and increase robustness against the process shifts. Networks of the mixed tree-mesh are especially good at minimizing the effects of manufacturing variations. Compared to completely tree-based topologies, mixed tree-mesh networks demand higher wirelength and routing resources. Because of the large mesh wiring, there may be a significant IR drop and increased power usage. Furthermore, creating such networks is more complicated and might not be appropriate for designs with limited resources [6].

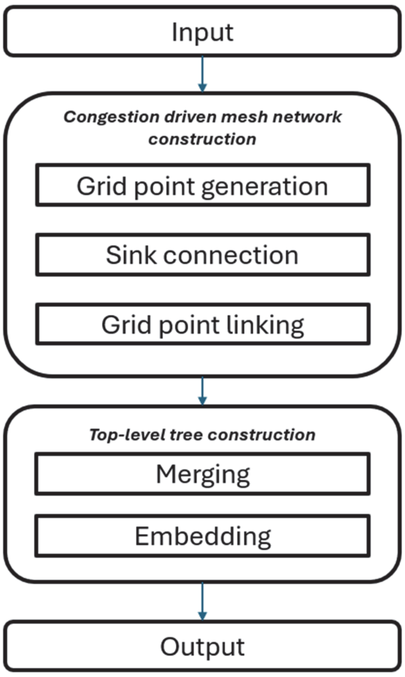


Fig. 5. The overview of mixed tree-mesh clock distribution networks

- DSO.ai – A distributed system to optimize the physical design flows.

In the method with DSO.ai, the distributed system has been built, for the optimization of the physical design flows, and correspondingly for CTS optimization. In the methodology, multiple iterations of parallel runs are used to optimize the enormous design parameters search space. The methodology provides a Pareto optimization opportunity, which opposes too long a runtime and a big CPU usage in terms of memory and cores [7,8].

Proposed method. To decrease the skew and fix the issues described in references, which are: routing congestion (result is short and opens), setup hold margins and runtime, machine learning especially diffusion models have been used. The main algorithm for the proposed method shown in Fig. 6.

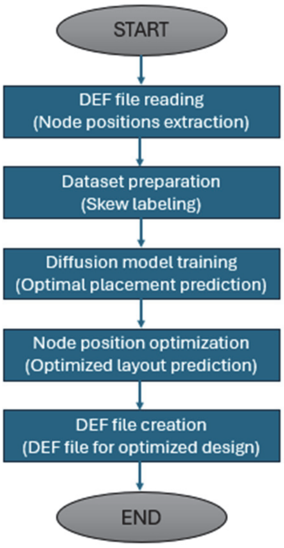
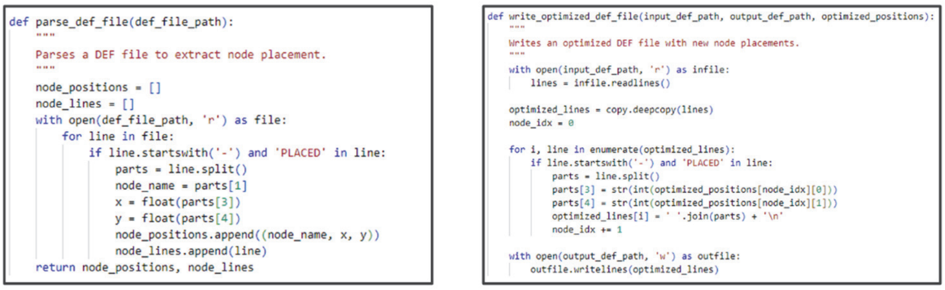

START
DEF file reading
(Node positions extraction)
Dataset preparation
(Skew labeling)
Diffusion model training
(Optimal placement prediction)
Node position optimization
(Optimized layout prediction)
DEF file creation
(DEF file for optimized design)
END

Fig. 6. The proposed method's algorithm

At the initial stages of optimization required to have the design's DEF file where CTS has been done without any optimization. During the DEF file reading process, the design's component placement has been extracted, at the end of the methodology output, a DEF file has been generated for the optimized design, both functions implemented with the help of python, and the implementation shown in Fig. 7.



```
def parse_def_file(def_file_path):
    """
    Parses a DEF file to extract node placement.
    """
    node_positions = []
    node_lines = []
    with open(def_file_path, 'r') as file:
        for line in file:
            if line.startswith('-') and 'PLACED' in line:
                parts = line.split()
                node_name = parts[1]
                x = float(parts[3])
                y = float(parts[4])
                node_positions.append((node_name, x, y))
                node_lines.append(line)
    return node_positions, node_lines
```

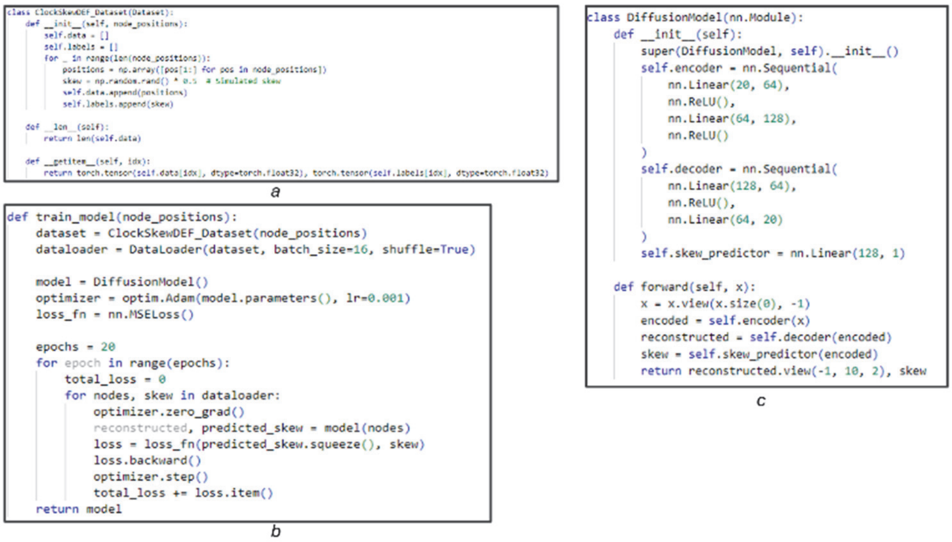
a)

```
def write_optimized_def_file(input_def_path, output_def_path, optimized_positions):
    """
    Writes an optimized DEF file with new node placements.
    """
    with open(input_def_path, 'r') as infile:
        lines = infile.readlines()

    optimized_lines = copy.deepcopy(lines)
    node_idx = 0

    for i, line in enumerate(optimized_lines):
        if line.startswith('-') and 'PLACED' in line:
            parts = line.split()
            parts[3] = str(int(optimized_positions[node_idx][0]))
            parts[4] = str(int(optimized_positions[node_idx][1]))
            optimized_lines[i] = ' '.join(parts) + '\n'
            node_idx += 1

    with open(output_def_path, 'w') as outfile:
        outfile.writelines(optimized_lines)
```

b)

Fig. 7. The input DEF reading (a) and output DEF (b) creation

For Dataset preparation, the instances initialize as a class, and there is generates random floats between 0 and 0.5 to simulate the skew, after this Diffusion model has been defined. Diffusion models are a class of generative models that transform simple noise distributions into complex data distributions through an iterative process. They work by simulating a forward process, which teaches to remove the noise step by step to generate realistic samples. The created model is designed to encode input data, reconstruct it and predict a skew value from the encoded representation. For diffusion model, a training function is created, be using a dataset of node positions. The training epocha number specified 20 and the initialized total loss for epochs specified as 0, the 20 is a common practice in machine learning, and 0 at the beginning of each epoch to accumulate the loss over all batches within that epoch. This helps in monitoring the training process by providing a measure of how well the model is performing after each epoch. All above-described functions are shown in Fig. 8.



```
class ClockSkewDEF_Dataset(Dataset):
    def __init__(self, node_positions):
        self.data = []
        self.labels = []
        for _ in range(len(node_positions)):
            positions = np.array([pos[1:] for pos in node_positions])
            skew = np.random.rand() * 0.5  # Simulated skew
            self.data.append(positions)
            self.labels.append(skew)

    def __len__(self):
        return len(self.data)

    def __getitem__(self, idx):
        return torch.tensor(self.data[idx], dtype=torch.float32), torch.tensor(self.labels[idx], dtype=torch.float32)
```

```
def train_model(node_positions):
    dataset = ClockSkewDEF_Dataset(node_positions)
    dataloader = DataLoader(dataset, batch_size=16, shuffle=True)

    model = DiffusionModel()
    optimizer = optim.Adam(model.parameters(), lr=0.001)
    loss_fn = nn.MSELoss()

    epochs = 20
    for epoch in range(epochs):
        total_loss = 0
        for nodes, skew in dataloader:
            optimizer.zero_grad()
            reconstructed, predicted_skew = model(nodes)
            loss = loss_fn(predicted_skew.squeeze(), skew)
            loss.backward()
            optimizer.step()
            total_loss += loss.item()
    return model
```

```
class DiffusionModel(nn.Module):
    def __init__(self):
        super(DiffusionModel, self).__init__()
        self.encoder = nn.Sequential(
            nn.Linear(20, 64),
            nn.ReLU(),
            nn.Linear(64, 128),
            nn.ReLU()
        )
        self.decoder = nn.Sequential(
            nn.Linear(128, 64),
            nn.ReLU(),
            nn.Linear(64, 20)
        )
        self.skew_predictor = nn.Linear(128, 1)

    def forward(self, x):
        x = x.view(x.size(0), -1)
        encoded = self.encoder(x)
        reconstructed = self.decoder(encoded)
        skew = self.skew_predictor(encoded)
        return reconstructed.view(-1, 10, 2), skew
```

Fig. 8. The dataset class (b), the diffusion model class (b) and the model training (c) implementation

At the end of optimization, the result is the skew improved design's DEF file, whose generation part is shown in Fig. 7b.

Results. To provide the best and worst sides of the proposed method 3 test designs have been implemented with different parameters and clock frequencies. The results are shown in Table below.

Table

Results

	Gate count	Clock frequency	Reference 1-3 methods			Design with DSO.ai			Proposed method		
			Runtime	Short count/ open count	Max skew	Runtime	Short count/ open count	Max skew	Runtime	Short count/ open count	Max skew
Design 1	~ 600K	8G	2.7h	15/2	0.252	26.4h	0/0	0.201	2.79h	1/0	0.198
Design 2	~ 1.5M	16G	46.03h	169/3	0.546	142.42h	141/1	0.430	47.5h	141/2	0.352
Design 3	~1.5M	20G	38.87h	78/1	0.601	121.28h	65/0	0.501	38.25h	67/1	0.402

Conclusion. Skew minimization is an important factor during the clock tree synthesis. Current CTS optimization methods have trade-off between the runtime, skew minimization and routing congestion. In the article, the skew minimization method with the use of diffusion models of machine learning. With the proposed method, the runtime has increased by ~5% in respond to EDA tools integrated methodologies, and decreased in terms of AI tool usage, as a result skew decreased by ~28%.

REFERENCES

1. **Micheli G. D.** Synthesis of Digital Circuits.- McGraw-Hill, 1994.
2. **Raju Gorla.** What is Clock Tree Synthesis (CTS), and why is it critical? - VLSI Web, November 29 2024.
3. **Peddi Anusha, K. Satish Babu**. Clock Tree Synthesis Analysis and Optimization in Physical Design Flow of Serial Peripheral Interconnect (SPI) // International Journal of VLSI System Design and Communication Systems.- October 2014.- P. 465-471.
4. **Anju Rose T., Gnana Sheela K.** A Survey on Buffered Clock Tree Synthesis for Skew Optimization // International Journal of Science and Research.- November 2014.- Vol. 3, issue 11.- P. 659-666.
5. Minimizing Skew and Delay with Buffer Resizing and Relocation during Clock Tree Synthesis / **P. Punia, Rouble, Shuka Kr. Neeraj, et al** // International Journal of Computer Applications.- June 2014.- Vol. 95 No.23.- P. 30-35.
6. Local Clock Skew Minimization Using Blockage-aware Mixed Tree-Mesh Clock Distribution Network / **L. Xiao, X. Zignag, Z. Qian, et //** Proceedings of the International Conference on Computer-Aided Design.- Nov. 2010.- P. 458-462.
7. **Piyush Verma**. DSO.ai - A Distributed System to Optimize Physical Design Flows // 2024 International Symposium on Physical Design (ISPD '24). Association for Computing Machinery.- New York, NY, USA, 2024.- P. 115–116, https://doi.org/10.1145/3626184.3639780
8. What is Design Space Optimization (DSO)? – How It Works? | Synopsys

National Polytechnic University of Armenia. “Synopsys Armenia” CJSC. The material is received on 14.01.2025.

Ա.Ա. ՂԱԶԱՐՅԱՆ

ՏԱԿՏԱՅԻՆ ԱԶԴԱՆՇԱՆԻ ՇԵՂՄԱՆ ՕՊՏԻՄԱԼԱՑՈՒՄԸ ԴԻՖՈՒԶԻՈՆ ՄՈԴԵԼՆԵՐԻ ԿԻՐԱՌՄԱՄԲ

Ինտեգրալ սխեմաների ֆունկցիոնալությունը զգալիորեն մեծանում է, ինչը ստիպում է ունենալ ավելի խիստ սահմանափակումներ և պահանջներ, հատակպես բարձր հաճախականային համակարգերում: Թվային ինտեգրալ սխեմաների նախագծման ավտոմատացված հոսքուղում սինքրոազդանշանի ծառի սինթեզը հանդիսանում է կարևոր կետերից մեկը, որն էականորեն ազդում է ինտեգրալ սխեմաների պարամետրերի վրա: Տակտային ազդանշանի շեղումը սինքրոազդանշանի ծառի սինթեզի որակական պարամետրերից է, որը պետք է հնարավորնիս փոքր լինի, հակառակ դեպքում՝ այն կազդի ինտեգրալ սխեմայի աշխատանքի վրա: Տակտային ազդանշանի շեղման նվազագույն արժեքի հայտնաբերումը ներկայացնում է միջմիացումների ուղիներ ծանրաբեռնվածության, ժամանակային պաշարների և մի շարք այլ պարամետրերի փոխզիջման կետ: Ներկայումս կան տարբեր մեթոդներ, որոնք ինտեգրված են ֆիզիկական նախագծման գործիքներում, որոնք նպատակ ունեն պարամետրերի միջև գտնել հնարավորինս օպտիմալ փոխզիջման կետեր: Որոշ ալգորթիմներ ընթացիկ գործընթացներում ճշգրիտ չեն, ինչը ստիպում է նախագծողին գործել ավտոմատացված հոսքուղուց դուրս: Նման խնդիրները շտկելու համար ներկայացվել է դիֆուզիոն մոդելների օգտագործմամբ տակտային ազդանշանի շեղումը նվազագույնի հասցնելու մեթոդ: Նպատակային մեթոդով գործարկման ժամանակն ավելացել է ~5%-ով՝ ի պատասխան EDA գործիքների ինտեգրված մեթոդաբանության, և նվազել է AI գործիքների առումով, իսկ շեղումը նվազել է ~28%-ով:

Առանցքային բառեր. սինքրոազդանշանի ծառի սինթեզ, տակտային ազդանշանի շեղում, ծրագծման ծանրաբեռնվածություն, մեքենայական ուսուցում, դիֆուզիոն մոդելներ

А.А. КАЗАРЯН

МЕТОД МИНИМИЗАЦИИ ПЕРЕКОСА С ИСПОЛЬЗОВАНИЕМ МОДЕЛЕЙ ДИФФУЗИИ

Функциональность интегральных схем (ИС) значительно возрастает, что заставляет предъявлять более строгие требования, особенно к быстрым системам. В цифровой методологии синтез дерева тактовых импульсов имеет решающее значение, что влияет на производительность ИС. Одним из параметров дерева тактовых импульсов является перекос тактовых импульсов, который должен быть минимальным, в противном случае - он повлияет на производительность ИС. Минимальное развитие перекоса - это компромисс между запасами по времени, перегрузкой маршрутизации и т.д. В настоящее время существуют различные методологии, интегрированные в инструменты физического проектирования, которые стремятся найти хорошие точки компромисса между параметрами. Некоторые из алгоритмов не работают должным образом в текущих процессах, и человек должен действовать вручную, что делает работу автоматизированных инструментов электронного проектирования неавтоматизированной. Для устранения таких проблем представлен метод минимизации перекоса с использованием моделей диффузии. При использовании целевого метода время выполнения увеличилось примерно на 5% в ответ на интегрированные методологии инструментов EDA и сократилось в отношении инструментов ИИ, а перекос уменьшился примерно на 28%.

Ключевые слова: синтез дерева синхросигнала, перекос, перегрузка маршрутизации, машинное обучение, модели диффузии.