Circuit-GNN: Graph Neural Networks for Distributed Circuit Design

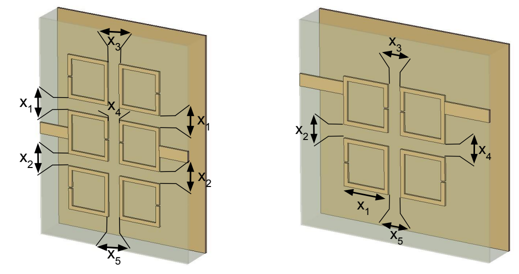
Guo Zhang [*1] **Hao He** [*1] **Dina Katabi** [1]

Abstract

We present Circuit-GNN, a graph neural network (GNN) model for designing distributed circuits. Today, designing distributed circuits is a slow process that can take months from an expert engineer. Our model both automates and speeds up the process. The model learns to simulate the electromagnetic (EM) properties of distributed circuits. Hence, it can be used to replace traditional EM simulators, which typically take tens of minutes for each design iteration. Further, by leveraging neural networks' differentiability, we can use our model to solve the inverse problem – i.e., given desirable EM specifications, we propagate the gradient to optimize the circuit parameters and topology to satisfy the specifications. We exploit the flexibility of GNN to create one model that works for different circuit topologies. We compare our model with a commercial simulator showing that it reduces simulation time by four orders of magnitude. We also demonstrate the value of our model by using it to design a Terahertz channelizer, a difficult task that requires a specialized expert. The results show that our model produces a channelizer whose performance is as good as a manually optimized design, and can save the expert several weeks of topology and parameter optimization. Most interestingly, our model comes up with new designs that differ from the limited templates commonly used by engineers in the field, hence significantly expanding the design space.

1. Introduction

Distributed circuit design refers to designing circuits at high frequencies, where the wavelength is comparable to or smaller than the circuit components. It is increasingly



(a) 6-resonator filter (b) 4-resonator filter

Figure 1. Examples distributed circuits templates based on square resonators. Each template has multiple control parameters like x_i.

important since new 5G and 6G communication technologies keep moving to higher and higher frequencies. Unlike lower frequencies, where there are relatively fast tools and a large literature on design guidelines, designing distributed circuits is a slow and onerous process. The process goes as follows. An expert engineer comes up with an initial design based on desired specifications (e.g., design a band-pass filter at 300 GHz, with a bandwidth of 30 GHz). To do so, typically the engineer picks an initial template and optimizes its parameters. Fig. 1 shows common templates used in the literature (Hong, 2011). The engineer then spends extensive effort optimizing the parameters of the template (the x_i's in Fig. 1) so that the circuit satisfies the desired specifications. The optimization is done iteratively. In every iteration, the engineer sets the parameters in the template to some values, simulates the design, and compares the output of the simulation to the specifications. Each simulation takes tens of minutes during which the simulator runs mainly a brute-force numerical solution of the Maxwell EM equations. This iterative effort may turn out to be useless if the chosen template topology cannot satisfy the desired specifications, and a new template must be tried and optimized using a new set of iterations. The process can take days, weeks, or months.

In this paper, we introduce a learning model that speeds up and automates this process. Our model addresses both the forward and inverse design problems. The forward task takes a circuit design and produces the resulting s_{21} function, which relates the signal on the circuit's output port to the signal at its input port. Said differently, the forward task produces the output of the EM simulator but using a neural network. The inverse task on the other hand takes

[*]Equal contribution [1]EECS, Massachusetts Institute of Technology, Cambridge, MA, USA. Correspondence to: Hao He <haohe@mit.edu>.

Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

the specifications, i.e., a desired s_{21} function, and produces a circuit that obeys the desired specifications. To solve the inverse task, we leverage that neural networks are differentiable. Thus, given a desirable s_{21} and an initial circuit, we back-propagate the gradient to optimize the circuit design so that it satisfies the desired specifications.

Designing a neural network model for distributed circuit design is challenging. The straightforward approach needs to train a different network for each template. This would be inefficient as clearly there is much shared information across templates. Furthermore, such an approach limits the design space to a specific set of templates, and prevents the model from attempting to design new templates.

To address these challenges, we develop a Graph Neural Network (GNN) model that works across a broad class of templates. We leverage that distributed circuits are typically designed using resonators as their building blocks. For example, it is common to use square resonators as in Fig. 1 (Hong & Lancaster, 1996), or ring resonators (Hong, 2011). By manipulating the number of resonators, their internal parameters, their relative distances, and orientations, one can design different circuits (Hong, 2011). Our approach leverages this property. We model the resonators in each circuit as nodes in a graph, and their electromagnetic coupling as edges between the nodes. This allows us to design a unified model that applies across templates. Apart from sharing information across templates, this design allows the network to produce new templates beyond the ones commonly used in the field. Designing a valid circuit however by back-propagation is non-trivial because nodes have to fit in a planner space and cannot overlap. To deal with these constraints we develop a novel multi-loop gradient descent algorithm with local re-parameterization.

This paper makes the following contributions:

- This paper is the first to present a deep learning model for circuit design that works across circuits with different templates, i.e., circuits that differ in the topology and number of basic components. All past papers train a separate neural network per template (Cao et al., 2009; Feng et al., 2016; 2017).
- The paper also presents the first deep learning model that solves the inverse distributed circuit design problem. Only (Zhang et al., 2018a) tried to solve the inverse problem, but their scheme works only for circuits with one parameter, and hence is not practical.[*]
- The paper introduces a novel approach for solving inverse optimizations over GNNs, where valid solutions have to maintain a planner graph with non-overlapping nodes. We do so via the re-parameterization in section 3.2.2. We believe this setup applies beyond circuit design to other graphs where nodes have a geometric and physical meaning.
- Finally, the paper proposes an alternative approach for combining the feature maps of the nodes to generate the feature map for the whole graph in a GNN. Unlike past work on GNN where the graph is represented using the sum or pooling operation, we propagate the information from the internal nodes to the input and output nodes, and represent the graph as a concatenation of the feature maps of the input and output nodes. We believe this is more suitable for graphs that interact with the rest of the world via specific inputs and outputs (e.g., circuits which has input and output ports). It also empirically works better for our GNN.

[*]The authors of (Zhang et al., 2018a) propose to map the specifications to an intermediate analytical representation called the "coupling matrix", but there is no general solution that maps a coupling matrix to an actual distributed circuit design.

We believe the paper makes an important leap towards learning circuit design by providing a practical solution that addresses relatively complex real-world circuit design problems.

2. Related Work

2.1. Learning-Based Circuit Design

Prior work on using machine learning for circuit design focuses on lumped design, where the circuit is represented as connections between lumped components: resistance, capacitance, inductance, transistor, etc. Such an approach however does not apply to high-frequency circuits, which require a distributed design. Further, these solutions train a separate model for each template and cannot work with unseen templates that were not used for training (Colleran et al., 2003; Liu et al., 2009; Lyu et al., 2018a;b;c; Wang et al., 2014; He et al., 2018). Some prior work has tried to do circuit optimization across topologies using genetic programming (McConaghy et al., 2011; Lourencco & Horta, 2012). These papers are different from ours both in terms of technical design and the fact that they target lumped circuits. They also face challenges including low convergence rate, slow optimization speed, and a difficulty in producing meaningful designs (Sorkhabi & Zhang, 2017).

Prior attempts at learning distributed circuit design have key limitations (Cao et al., 2009; Feng et al., 2016; 2017). On the one hand, each of their trained models works only with a particular template and parameter setting, while our GNN works across templates. On the other hand, unlike our model which directly predicts the values of the transfer function, s_{21}, for different frequencies, they predict a parameterized version of the s_{21} function. Specifically, they leverage that the transfer function can be approximated as $s_{21}(\omega) = \sum_{i=1}^{N} \frac{a_i}{j\omega - b_i}$, where ω is the frequency and a_i and b_i are complex parameters. Thus,instead of predicting the function

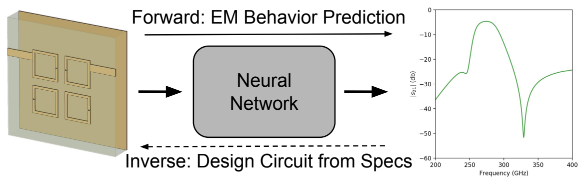


Figure 2. Illustration of the forward and inverse models.

$s_{21}(\omega)$, they train a neural network to predict the parameters a_i and b_i. This approach has several limitations. First, given a particular template even without any parameter setting, it is not clear how many parameters, a_i and b_i, one needs to have for a good representation of the transfer function. Thus, past work, for each template, trains multiple neural networks. Second, during inference, it is not clear how to pick the best network for a particular template. Thus, past work trains an additional model that selects which network to use from the set of networks associated with that template.

2.2. Graph Neural Networks for Relation Modeling

Graph neural network (GNN) is a well-known architecture for modeling relations. Researchers have used GNN to model interaction between physical objects (Watters et al., 2017; Zheng et al., 2018; Battaglia et al., 2016), chemical bonds between atoms (Jin et al., 2017; 2018), and social networks (Perozzi et al., 2014). GNN has also been used in computer vision for modeling the human skeleton for activity recognition (Zhang et al., 2018b) and person re-identification (Shen et al., 2018). We are inspired by these prior work; however we apply GNN to a new domain where the nodes in the graph are circuit components, and the relationships between them stem from electrical and magnetic coupling. We leverage our domain knowledge to customize GNN to our problem and ensure the model capture the underlying electromagnetics and produces valid circuits.

3. Learning Distributed Circuit Design

Figure 2 provides a high-level abstraction of our model. In the forward direction, the model maps a given circuit to the corresponding transfer function (i.e. s_{21}). In the inverse direction, the model uses gradient descent to optimize the circuit topology and parameters to produce a desired s_{21}.

In the context of distributed circuit design, circuits have a geometric representation as illustrated in the left panel of Figure 2 or the templates in Figure 1. Typically, these circuits are compositions of a parametrizable building block. Common building blocks include square resonators (Hong & Lancaster, 1996) like the ones in Figure 1, ring resonators (Hong, 2011), or bar resonators (Levy et al., 2002). The behavior of the circuit, i.e., its transfer function, depends on how each resonator is parametrized, the distance between the resonators, and their relative orientation. For example, the square resonators in Figure 1 may have slightly different sizes. They also have a slit that can be facing up, down, left, or right. Further, they are separated by different gaps which affect their coupling behavior. For clarity, we will describe our design in the context of square resonators, as the key building block. Our approach, however, is general and applies to circuits built using other resonators or building blocks.

3.1. Forward Model

The goal of the forward model is to allow the circuit designer to quickly obtain the transfer function of his/her design. It takes as input the geometric representation of the circuit, and outputs a complex-valued vector that provides a discrete representation of the circuit transfer function s_{21}.

3.1.1. Model

Figure 3 shows a detailed description of the forward model. The figure shows that the model has four steps, which we describe below.

STEP 1: From Circuit to Graph. In the first step, we map the circuit geometric representation to a graph, where each node refers to a resonator and each edge refers to the interaction (i.e., the electromagnetic coupling) between a pair of resonators.

A circuit having N square resonators has N raw parameter vectors. Each square resonator has a parameter vector $[x, y, a, \theta]^T$ as shown in Figure 4(a), where (x, y) is the center position of the square, a is the side length of the square, and θ is the angular position of the slit. From this raw input, we generate a graph circuit representation with node attributes and edge attributes. Node i has attributes $\mathbf{n}_i$ containing $[a_i, \theta_i]^T$ which is a subset of the raw parameters. Notice that the resonator center position information are not provided to the node because the absolute positions of the circuit is meaningless. Edge attributes $\mathbf{e}_{ij}$ between node i and node j contain $[\theta_i, \theta_j, x_i - x_j, y_i - y_j, g_{ij}, s_{ij}]^T$ where $x_i - x_j$ and $y_i - y_j$ are the relative position of the two components. g_{ij} and s_{ij} indicate the length of the *gap* and *shift* between two square resonators as demonstrated in Figure 4(b). Gap and shift informations are included in the edge attributes because they are important to infer the coupling coefficient between two resonators (Hong & Lancaster, 1996). Note that not all pairs of nodes in the graph have an edge. Since electromagnetic coupling decays with distance, we leverage the data in (Hong, 2011) to set a threshold on distance beyond which two resonators do not share an edge. (See graphs in Figure 5.)

STEP 2: Graph Encoding. In this step, we use GNN to extract the nodes' and edges' features. We process the graph circuit input using a k-layer graph neural network. The t^{th} GNN layer has two sub-nets *node processor* f_n^t and *edge*

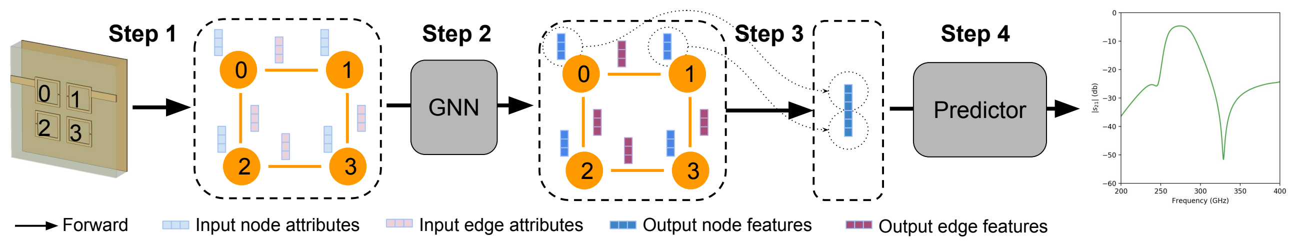
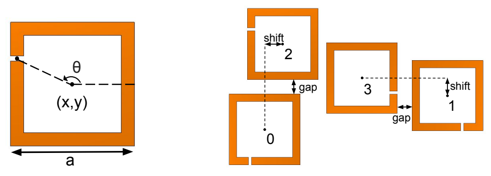


Figure 3. **Forward Model.** Step 1 maps circuit geometry to a graph; step 2 performs graph encoding using a GNN; step 3 generates a global representation of the original circuit, finally step 4 predicts the circuit's complex-valued transfer function.



(a) Resonator (b) Gap and Shift

Figure 4. **Data contributing to node and edge attributes.** (a) shows a resonator which is described by 4 parameters: its center position (x, y), the angular position of its open slit θ, and its edge length a; (b) shows the definition of gap and shift in our setting.

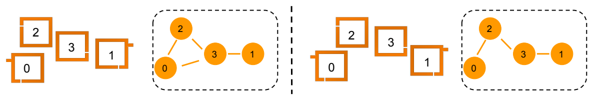
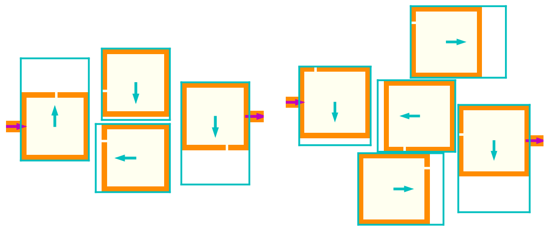


Figure 5. **Graph examples.** Figure shows two circuits and the resulting graphs. Note that the edge between node 0 and node 3 disappears when they are too far from each other.

processor f_e^t. The input node features $\boldsymbol{\eta}_i^{t-1}$ are transformed to outputs as follows. First we compute the coupling effect that every resonator receives from its neighbors. We use $\boldsymbol{\epsilon}_{ij}^t$ to denote the coupling effect imposed on resonator i by its nearby resonator j. For this edge effect $\boldsymbol{\epsilon}_{ij}^t$, we call i and j the receiver node and sender node respectively. We use the edge processor f_e^t to learn this effect. Its input is the concatenation of receiver node features, sender node features and the edge attributes. Mathematically, we can calculate the edge effect from node j to node i as follows, $\boldsymbol{\epsilon}_{ij}^t = f_e^t(\boldsymbol{\eta}_i^{t-1}, \boldsymbol{\eta}_j^{t-1}, \mathbf{e}_{ij}^{t-1})$. Second we sum all the edge effects at each node as its total received coupling effect. Then the node processor is used to update the node features based on its current node features and the total coupling effect this node received. Thus the new features for node i, $\boldsymbol{\eta}_i^t$ can be calculate as $\boldsymbol{\eta}_i^t = f_n^t(\boldsymbol{\eta}_i^{t-1}, \sum_j \boldsymbol{\epsilon}_{ij}^t)$. In implementation, we model node and edge processors f_n^t, f_e^t as Multilayer perceptrons (MLP) with LeakyReLU activations.

STEP 3: Graph Summarization. In this step, we transform all graph node features into a *fixed length* global graph features $\mathbf{g}$. To do so, we concatenate the features of two special nodes that represent the components that connect the circuit to the input and output ports. Mathematically, $\mathbf{g} \triangleq [\boldsymbol{\eta}_0^k, \boldsymbol{\eta}_1^k]$ since we have manually indexed the square resonators connected to input/output ports as 0 and 1.



(a) 4-resonator (b) 5-resonator

Figure 6. Visualization of the allowed moving ranges for each resonator in the circuit during one optimization step. The cyan box indicates the area where the resonator could move, and the cyan arrow shows the sampled direction during that optimization step.

STEP 4: Prediction. Finally, in the last stage, we use the prediction network to predict the circuit's transfer function. The prediction network outputs the prediction $\hat{\mathbf{y}}$ of the s_{21} parameter for the circuit graph $\mathbf{g}$. Since the transfer function is complex-valued we design the prediction network as a multi-layer fully connected network with residual links and complex-valued weights. The output is a complex-valued vector $\hat{\mathbf{y}}$ that provides a discrete representation of the circuit transfer function, i.e., $\hat{\mathbf{y}} \triangleq [s_{21}(\omega_1), \cdots, s_{21}(\omega_m)]^T$ where $\{\omega_i\}_{i=1}^m$ indicates the frequency samples from the circuit working frequency $\Omega \triangleq [\omega_{min}, \omega_{max}]$.

The model represented by the above four steps is a composition of neural networks, and hence it can be represented as an end-to-end neural network. Denoting the raw circuit parameter as $\mathbf{r} \in \mathbb{R}^{N\times 4}$, our model is taught to capture the relationship $\hat{\mathbf{y}}(\mathbf{r})$ between circuit parameters $\mathbf{r}$ and the true (discretized) transfer function $\mathbf{y} \in \mathbb{C}^m$.

3.1.2. TRAINING

We use the conventional supervised learning paradigm to train our model. During training, l_1 loss are used. The loss function is $\mathcal{L}(\theta) \triangleq \mathbb{E}_{(\mathbf{r},\mathbf{y})\sim\mathcal{D}} \|\Re(\hat{\mathbf{y}}(\mathbf{r};\theta) - \Re(\mathbf{y})\|_1 + \|\Im(\hat{\mathbf{y}}(\mathbf{r};\theta) - \Im(\mathbf{y})\|_1$, where $\mathcal{D}$ is the dataset containing all traing samples $(\mathbf{r}, \mathbf{y})$.

3.2. Inverse Optimization

We describe how we use the forward neural network model to solve the inverse problem and obtain a circuit design that satisfies a given transfer function.

3.2.1. Optimization Objective

Our model's ability to solve inverse design problem is automatically gained by the differentiable nature of the neural network. For any differentiable objective function $\mathcal{J}(\hat{\mathbf{y}})$ over the transfer function $\hat{\mathbf{y}}$, we can apply gradient descent methods to optimize the input parameters of the neural network. In particular, consider a designer who wants to obtain a circuit geometry that satisfies a desired transfer function $\mathbf{y}^*$. We can define the objective function as the l_2-norm of the difference between the desired transfer function and that delivered by our design, i.e. $\mathcal{J}(\hat{\mathbf{y}}) \triangleq \|\hat{\mathbf{y}}(\mathbf{x}) - \mathbf{y}^*\|^2$. The choice of the objective function can also change depending on the design. For example, when designing a band-pass filter, the goal is to allow signals in specific frequency bands to pass through, and block signals outside the desired bands.

In this case, we set the objective function to be the squared distance between the circuit transfer function and an ideal one as shown in Eqn. 1.

$$\mathcal{J}(\hat{\mathbf{y}}) = \sum_{i:\omega_i \in \Omega^*} (|\hat{y}_i| - 1)^2 + \sum_{i:\omega_i \notin \Omega^*} |\hat{y}_i|^2 \tag{1}$$

Here, Ω^* indicates the union of all pass-bands. An ideal transfer function would satisfies $|\mathbf{y}(\omega)| = 1_{[\omega \in \Omega^*]}$ i.e. blocking all signals outside the pass-band Ω^*.

3.2.2. Re-parameterization

Unfortunately, solving the inverse problem simply by back propagating the gradient may lead to an invalid circuit design. The feasible set of the solution space (the space for all valid circuits) is not convex. For example, in a circuit, Component A could be on the left of Component B or the right of Component B. But if we interpolate the two cases, component A and Component B will overlap resulting in an invalid circuit. Thus, we need to design a gradient projection mechanism to make sure that during the optimization the circuit will never be outside the space of valid circuits. Or, alternatively, we can re-parameterize the circuit such that all solutions in the new parameter space are guaranteed to be valid.

Among all raw input parameters, only the change of resonator center positions may cause an invalid circuit. To address this issue, we introduce a novel re-parameterization to the center positions $\{\mathbf{p}_i = (x_i, y_i)\}_{i=1}^N$. First, we make a constraint that in each optimization step, a resonator can move along only one of the four directions: left, right, up and down. To do this, we uniformly sample a direction vector $\mathbf{d}_i$ for each resonator from the space {(-1,0),(1,0),(0,-1),(0,1)}. Second, we compute the maximum distance m_i that a resonator can move along the chosen direction. The maximum distance is decided to ensure that

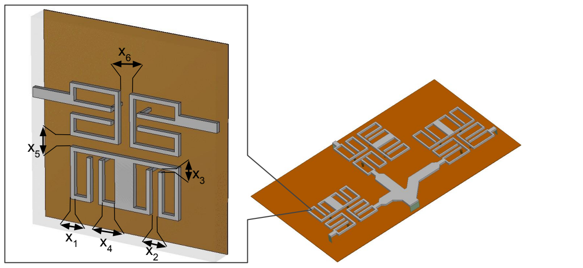


Figure 7. The schematic of a THz channelizer. It is designed by an human expert using weeks. It contains 3 submodule filters at different frequencies.

no two resonators collide even if all resonators move the furthest allowed. We then re-parameterize the center position of a resonator $\mathbf{p}_i$ by the parameter $q_i \in \mathbb{R}$ as follows, $\mathbf{p}_i = \mathbf{p}_i^* + \sigma(q_i) m_i \mathbf{d}_i$ where $\sigma(\cdot)$ is the sigmoid function and $\mathbf{p}_i^*$ is the original center position of the resonator. Since the sigmoid function is limited between 0 and 1, this re-parameterization ensures that the center position of a resonator does not move by more than the maximum allowable distance m_i in the direction $\mathbf{d}_i$. Figure 6 shows two random examples of acceptable moving ranges for all resonators in the circuit during one optimization step. Notice that although in each optimization step the resonator can only move in one direction, in the longer run, a resonator is still able to go everywhere by zig-zaging.

4. Experimental Evaluations

4.1. Distributed Circuits

Templates with square resonators. We experiment with a broad class of distributed circuits templates based on open-loop square resonators. Each such circuit is a composition of a number of square resonators with different width, relative location, and orientation. As illustrated in Figure 4, each resonator is characterized by four parameters, its center position (x, y), the angular position of its open slit θ, and the square width a. These templates are designed and simulated on the metal layers of an integrated chip based on the IHP SG13G2 process (IHP, 2018), a real world high frequency IC platform. The operation frequency of these circuits varies between 200 GHz to 400 GHz. In our experiments, during the data generation stage, we force all the resonators in one circuit to have an equal width a. The range of a is set to $[50\mu\text{m}, 75\mu\text{m}]$. Resonator center position (x_i, y_i) is sampled in a manner that fulfill the constraint that the gap between two nearby resonators should be in a reasonable range such as $[\frac{a_i}{80}, \frac{a_i}{5}]$. The slit position θ of each resonator is independently sampled from $\mathcal{U}[0, 2\pi)$. As for the circuit topology, it is sampled from some predefined topology patterns. An overview of all topologies in our dataset[†] is visualized in supplementary.

[†]For more dataset details, please see our project website: https://circuit-gnn.csail.mit.edu.

Table 1. **Performance of forward learning on circuits with different number of resonators.** Number of training samples are also listed here. Three-resonator and six-resonator circuits are only included in testset. The table shows that the model generalizes well even to circuit topologies and sizes that never seen in training.

# of Resonators	# of Samples	Training Error	Test Error
3	5790	-	1.423 dB
4	46423	0.923 dB	1.386 dB
5	37488	1.097 dB	1.785 dB
6	5859	-	2.552 dB

THz channelizer. Designing a Terahertz circuit is very challenging. We consider a real-world THz channelizer, which is used for a 100-Gbps-level chip-to-chip communication IC. The channelizer is designed by a senior Ph.D. student in the Terahertz research group in our department. This design took him weeks and involved simulations on a supercomputer. It is designed on the same IC plateform as we are using. The channelizer operates from 200 GHz to 400 GHz and has three channels centered at 235, 275 and 315 GHz, each having a bandwidth of 30 GHz. As shown in figure 7, the channelizer has 3 sub-modules that correspond to the three channels. In this work, we are going to show our model can actually design a THz chanelizer using square resonators which outperforms the human design for the desired specifications.

4.2. Dataset and Training Protocol

To train our network, we generate labeled examples using the CST STUDIO SUIT (CST official website, 2018), a commercial EM simulator. Generally, it takes about 10 to 50 CPU minutes to simulate one circuit. We generate about 100,000 circuit samples made of 3 to 6 resonators on a distributed computing cluster with 800 virtual CPU cores. We train on 80% of the data with 4 and 5 resonators, and test on the rest, including the data with 3 and 6 resonators which are 100% reserved for testing. Training uses the Adam optimizer (Kingma & Ba, 2014) and a batch-size of 64. In total, the model is trained 500 epochs. The learning rate is initialized as 10^{-4} and decayed every 200 epochs by a factor of 0.5.

One may wonder whether training on simulated circuits can capture the real-world circuits. Unlikely simulators in other fields, circuits simulators are highly accurate for two reasons. First, the manufacturing process can generate a circuit that accurately matches the simulated geometry and material. Second, the simulator solves the Maxwell equations which capture the exact physics with no approximation.

4.3. Evaluation of the Forward Task

We train our model on circuits with 4 and 5 resonators and test on circuits with 3, 4, 5, and 6 resonators. For each test case, we compute the error as the mean value of the absolute difference between the magnitude of the predicted s_{21} parameter $\hat{\mathbf{y}}$ and the ground truth $\mathbf{y}$ in dB, i.e. $\epsilon_{db}(\hat{\mathbf{y}}, \mathbf{y}) \triangleq \frac{1}{m}\sum_{i=1}^{m} |20\log_{10}(|y_i|) - 20\log_{10}(|\hat{y}_i|)|$. As common in circuit literature, we express the error in the dB domain.

Table 1 shows the evaluation results of the forward model prediction. The table shows that our model achieves a mean training error of 0.92dB, and 1.10dB on circuits with four and five resonators respectively. Such an error is fairly small, indicating that a single model is able to fit the data and captures the electromagnetic properties of both 4-resonator and 5-resonator circuits.

In terms of generalization, we consider both the model's ability to generalize to circuits with the same number of resonators as those it trained on (4, and 5 resonators), and circuits with different numbers of resonators that the model never seen (3 and 6 resonators). We refer to these two cases as same-topology-size and different-topology-size. The table shows that for topologies of 4 and 5 resonators, the test error is 1.40dB and 1.79dB. While these errors are larger than the training errors, they are still reasonably good. We believe that the difference in error between training and testing is due to limited data. With a larger training dataset, the error can be smaller.

A key characteristic of our GNN mode is its ability to generalize to new topologies with a different number of resonators than those used in training. Table 1 shows that error on three-resonator circuits are about the same as the errors on four or five-resonator circuits, while the error of six-resonator circuits is slightly higher than that of four and five resonators. Since an error of a few dB at such high frequencies is still reasonable, we believe our model does learn the relational effects between resonators and has the ability to generalize to new circuits templates unseen in the training.

For qualitative evaluation, we visualize randomly sampled prediction results for all kinds of circuits in Figure 8. Every row in the figure corresponds circuits with the same number of resonators. In each raw, we display two circuits with different topologies to illustrate that our model is robust to topology variations. As we can see, the model predicts accurate s_{21} parameter. Some tiny error appears in the less important frequency range in the stopband which are far from the passband of the filter. Prediction results on six resonator circuits is the most difficult. Although not exactly recovering the s_{21}, our model still accurately predicts the peak position and the filter cut-offs.

In terms of the run-time for prediction, our model conducts one prediction in 50 milliseconds on a single NVIDIA 1080Ti GPU which is four orders of magnitude faster than running one simulation using CST on a modern desktop.

Finally, we show example cases where the model fails and

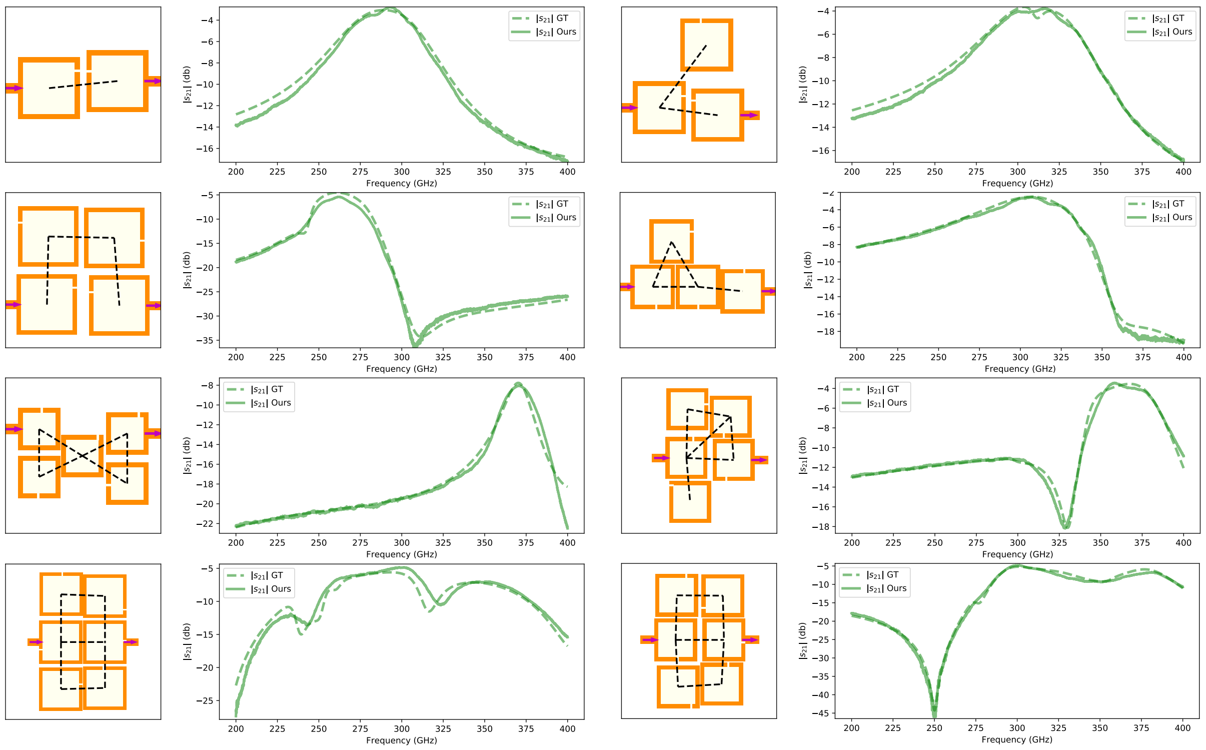


Figure 8. **Qualitative results of the forward model:** In each of the eight plots, the circuit is drawn on the left and the s_{21} are plotted on the right. The dashed lines and solid lines indicate the ground truth and our prediction respectively. In the circuit pictures, the position of input/output ports are indicated by two yellow arrows. We use black dashed lines to highlight the resonator pairs that are close enough to have relatively strong coupling effects.

generates large prediction error. For further analysis, we visualize some bad examples in Figure 9. In these cases, our model outputs the right trend of the s_{21} parameters, but mistakenly predicts the position of peaks (upside and downside) or predicts a wrong peak value.

Overall we believe these results show a significant leap in learning distributed circuit design. They enable engineers to quickly simulate their designs to ensure that they are within a few dB of the desired specifications. If more accuracy is desired, the engineers may then fine tune the final design using a commercial simulator. We will next show that the model can be used to automatically generate new circuit designs and expand the design space.

4.4. Evaluation of the Inverse Optimization

We evaluate our model's ability to solve the inverse design problem by comparing it with both a human expert and commercial software.

4.4.1. Comparison with Human Expert

We compete with a human expert on the task of designing the Terahertz channelizer described in section 4.1. We design each sub-modules of the channelizer individually as the human expert did. To design a sub-module which basically is a uni-band filter, we employ the filter design objective function introduced in Eqn. 1. We start from 200 random initialized circuits with different number of resonators and topologies. We then iteratively apply the local reparameterization and gradient descent steps 5000 times to optimize those circuits in parallel. At the end, we pick the circuit that produces the best objective value as our inverse design result. The whole process is fast and completes in less than 2 minutes on a single GPU.

We compare the transfer function produced by our model with the transfer function of the channelizer designed by the human expert using the CST simulator. Figure 10 shows the transfer functions for both designs. Recall that the channelizer is required to have three bandpass filters, which we highlight with the shaded regions in the figure. Generally, the frequency region where the transfer function is over -6dB is considered as the pass-band of the circuit. By comparing the expert design with the automated design from our model, we see that both designs are centered perfectly in the desired frequency bands. Further, by looking at the area above the -6dB line, we see that the filter designed by our model has better insertion loss –i.e., it delivers more power in the desired bands. In the supplementary material, we show the circuit designed by the model, which is different from the human picked circuit shown in Figure 7. This

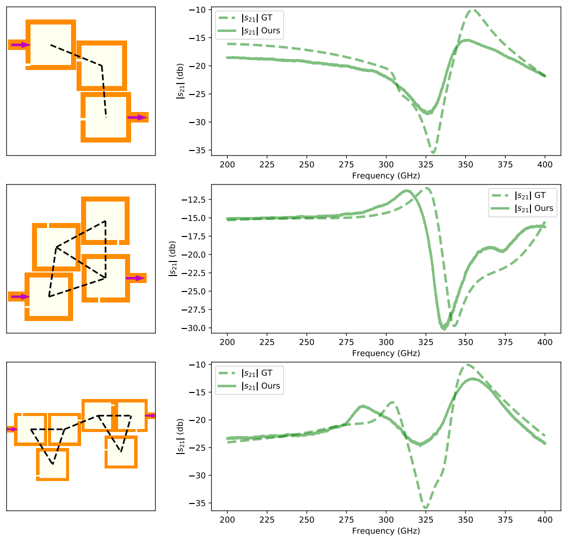


Figure 9. **Failure examples:** These examples have relatively large error, however, they still capture the trend of the s_{21} curve.

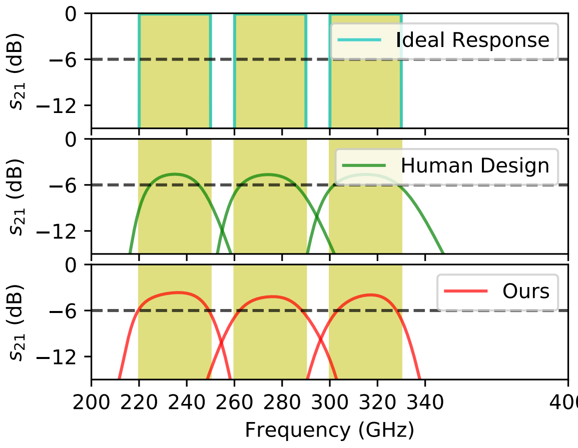


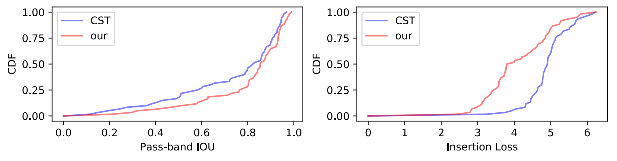
Figure 10. The transfer function of our optimized channelizer vs. the channelizer designed by the human expert. The yellow area indicate the desired pass-bands. Generally, the frequency region where the transfer function is over $-6dB$ is considered as the pass-band. The difference from 0dB is the insertion loss.

shows that the model has figured out a new circuit design that differs from the one produced by the human expert.

Finally, we note that the human expert spent several weeks optimizing the parameters in the channelizer's template to obtain his design, This indicates that the model could save the expert weeks of work. Even if the expert may not use the output of the model as is, starting from such a close design can save the expert much time and effort and this method is always fail-safe.

4.4.2. Comparison with Commercial Software

Commercial circuit simulators provide automatic parameter tuning tools that engineers may use to tune their design. In this section, we compare our model with CST parameter tuning, for the task of designing a uni-band bandpass filter. The



(a) CDF of Pass-band IOU (b) CDF of Insertion Loss

Figure 11. Our model versus commercial software (CST) for circuit design. Our model (in red) has higher pass-band IOU and lower insertion loss than CST (in blue).

filter bandwidth and center frequency are randomly chosen in $[20, 40]$ GHz and $[235, 315]$ GHz respectively. CST is allowed to run 450 simulations to search for a suitable design using the particle swarm method. In total, each CST design takes about 8 hours. In contrast, our model proposes only 10 design candidates. Running simulations to verify the candidates and pick the best one takes around ten minutes.

To evaluate the resulting filters, we use two metrics corresponding to two crucial filter properties: **Pass-band IOU** is used to measure how close the pass-band of the circuit is to the target band. Specifically, the pass-band of a circuit is defined as the frequency region where the transfer function magnitude is higher than its maximum magnitude minus 3db. Assume the circuit's pass-band is $\Omega = [\omega_L, \omega_R]$ and the pass-band wanted is Ω^*, the pass-band IOU is the intersection over union between Ω and Ω^*, i.e $\frac{|\Omega \cap \Omega^*|}{|\Omega \cup \Omega^*|}$. **Insertion Loss** is defined as the negative of the maximum transfer function magnitude. The smaller the insertion loss is, the more power the circuit delivers in the desired bands, and hence the better the circuit is.

The results shown in figure 11 demonstrate that the circuits produced by our model have better pass-band IOU and insertion loss. On average across all 60 design tasks, the circuits produced by our model have 0.80 pass-band IOU and 4.13db insertion loss while the circuits delivered by CST have 0.73 pass-band IOU and 4.92db insertion loss.

5. Conclusion

We present a powerful graph neural network model that can be used both for simulating distributed circuits, and automating the design process. The key property of the model is its ability to capture various circuits that differ in their topologies and sizes. We show that the model can reduce circuit simulation time by four orders of magnitude, while maintaining reasonable accuracy. We believe engineers can use the model to quickly hone in on a good design. They may then leverage a commercial simulator for final tuning. We also show that the model generates new circuit designs to match desired specifications. Interestingly these generated circuits are intrinsically different from the regular standard templates which confine today's designs. This capability is helpful for both automating circuit design and also expanding the design space.

References

Battaglia, P., Pascanu, R., Lai, M., Rezende, D. J., et al. Interaction networks for learning about objects, relations and physics. In *Advances in neural information processing systems*, pp. 4502–4510, 2016.

Cao, Y., Wang, G., and Zhang, Q.-J. A new training approach for parametric modeling of microwave passive components using combined neural networks and transfer functions. *IEEE Transactions on Microwave Theory and Techniques*, 57(11):2727–2742, 2009.

Colleran, D. M., Portmann, C., Hassibi, A., Crusius, C., Mohan, S. S., Boyd, S., Lee, T. H., and del Mar Hershenson, M. Optimization of phase-locked loop circuits via geometric programming. In *Custom Integrated Circuits Conference, 2003. Proceedings of the IEEE 2003*, pp. 377–380. IEEE, 2003.

CST official website. Cst - computer simulation technology. https://www.cst.com/, 2018. [Online].

Feng, F., Zhang, C., Ma, J., and Zhang, Q.-J. Parametric modeling of em behavior of microwave components using combined neural networks and pole-residue-based transfer functions. *IEEE Transactions on Microwave Theory and Techniques*, 64(1):60–77, 2016.

Feng, F., Zhang, C., Ma, J., Zhang, Q.-J., et al. Parametric modeling of microwave components using adjoint neural networks and pole-residue transfer functions with em sensitivity analysis. *IEEE Transactions on Microwave Theory and Techniques*, 65(6):1955–1975, 2017.

He, H., Zhang, G., Holloway, J. W., and Katabi, D. End-to-end learning for distributed circuit design. In *Workshop on ML for Systems at NeurIPS*, 2018.

Hong, J.-S. *Microstrip Filters for RF/Microwave Applications*, volume 235. John Wiley & Sons, 2011.

Hong, J.-S. and Lancaster, M. J. Couplings of microstrip square open-loop resonators for cross-coupled planar microwave filters. *IEEE Transactions on Microwave theory and Techniques*, 44(11):2099–2109, 1996.

IHP. Ihp low-vol & multi-project service. https://www.ihp-microelectronics.com/, 2018. [Online].

Jin, W., Coley, C., Barzilay, R., and Jaakkola, T. Predicting organic reaction outcomes with weisfeiler-lehman network. In *Advances in Neural Information Processing Systems*, pp. 2607–2616, 2017.

Jin, W., Barzilay, R., and Jaakkola, T. Junction tree variational autoencoder for molecular graph generation. *arXiv preprint arXiv:1802.04364*, 2018.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Levy, R., Snyder, R. V., and Matthaei, G. Design of microwave filters. *IEEE Transactions on Microwave Theory and techniques*, 50(3):783–793, 2002.

Liu, B., Wang, Y., Yu, Z., Liu, L., Li, M., Wang, Z., Lu, J., and Fernández, F. V. Analog circuit optimization system based on hybrid evolutionary algorithms. *INTEGRATION, the VLSI journal*, 42(2):137–148, 2009.

Lourencco, N. and Horta, N. Genom-pof: multi-objective evolutionary synthesis of analog ics with corners validation. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, pp. 1119–1126. ACM, 2012.

Lyu, W., Xue, P., Yang, F., Yan, C., Hong, Z., Zeng, X., and Zhou, D. An efficient bayesian optimization approach for automated optimization of analog circuits. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 65(6): 1954–1967, 2018a.

Lyu, W., Yang, F., Yan, C., Zhou, D., and Zeng, X. Batch bayesian optimization via multi-objective acquisition ensemble for automated analog circuit design. In *International Conference on Machine Learning*, pp. 3312–3320, 2018b.

Lyu, W., Yang, F., Yan, C., Zhou, D., and Zeng, X. Multi-objective bayesian optimization for analog/rf circuit synthesis. In *Proceedings of the 55th Annual Design Automation Conference*, pp. 11. ACM, 2018c.

McConaghy, T., Palmers, P., Steyaert, M., and Gielen, G. G. Trustworthy genetic programming-based synthesis of analog circuit topologies using hierarchical domain-specific building blocks. *IEEE Transactions on Evolutionary Computation*, 15(4):557–570, 2011.

Perozzi, B., Al-Rfou, R., and Skiena, S. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710. ACM, 2014.

Shen, Y., Li, H., Yi, S., Chen, D., and Wang, X. Person re-identification with deep similarity-guided graph neural network. In *European Conference on Computer Vision*, pp. 508–526. Springer, 2018.

Sorkhabi, S. E. and Zhang, L. Automated topology synthesis of analog and rf integrated circuits: A survey. *Integration, the VLSI Journal*, 56:128–138, 2017.

Wang, Y., Orshansky, M., and Caramanis, C. Enabling efficient analog synthesis by coupling sparse regression and polynomial optimization. In *Design Automation Conference (DAC), 2014 51st ACM/EDAC/IEEE*, pp. 1–6. IEEE, 2014.

Watters, N., Tacchetti, A., Weber, T., Pascanu, R., Battaglia, P., and Zoran, D. Visual interaction networks. *arXiv preprint arXiv:1706.01433*, 2017.

Zhang, C., Jin, J., Na, W., Zhang, Q.-J., and Yu, M. Multi-valued neural network inverse modeling and applications to microwave filters. *IEEE Transactions on Microwave Theory and Techniques*, 66(8):3781–3797, 2018a.

Zhang, X., Xu, C., and Tao, D. Graph edge convolutional neural networks for skeleton based action recognition. *arXiv preprint arXiv:1805.06184*, 2018b.

Zheng, D., Luo, V., Wu, J., and Tenenbaum, J. B. Unsupervised learning of latent physical properties using perception-prediction networks. *arXiv preprint arXiv:1807.09244*, 2018.