FASCINET: A Fully Automated Single-Board Computer Generator Using Neural Networks

Morteza Fayazi, *Graduate Student Member, IEEE,* Zachary Colter, Zineb Benameur-El Youbi, Javad Bagherzadeh, Tutu Ajayi, Ronald Dreslinski, *Senior Member, IEEE*

***Abstract*—Designing Single-Board Computers (SBCs) is becoming more challenging given the growing number of discrete components that are made available and the rate at which this number grows. Keeping track of all available components options, revisions, and functionalities is challenging for SBC designers who are striving for faster design cycles. Moreover, the procedure of deciding peripheral components, their values, and connections of an SBC is not only difficult because of various parameters that need to be considered, but also is time-consuming as there exist numerous components on a typical SBC nowadays. In this paper, an SBC generator tool, FASCINET, is presented that uses a Neural Network (NN) model to design customized peripheral circuits for SBCs. The tool creates a large Commercial Off-the-Shelf Database (COTS DB) of existing components, efficiently searches through them, and selects optimal components for both main and peripheral components based on the user's requirements.**

Creating such a broad COTS DB requires processing abundant datasheets. A manual approach is time-consuming, even if only a fraction of all available datasheets is considered. In order to automate this process, this paper describes a novel NN-based approach for automatically categorizing datasheets and proposes an extraction technique for parsing relevant functional information from tables within. Our evaluation using a test set that contains over $770{,}000$ components shows that the category of datasheets is identified correctly over 95% of the time. Additionally, the table extractor has a precision above 96%. Our proposed fully autonomous SBC design approach reduces the time for generating the schematic of an SBC to as little as two minutes. For validating the accuracy of our model, the netlists of 400 SBCs designed by FASCINET are compared to the human-designed versions. This evaluation shows that FASCINET is able to design SBCs that are identical to the manually-designed ones except for minor differences.

***Index Terms*—Single-board computer, automatic peripheral circuit generator tool, neural network, COTS DB, datasheet, data mining, category recognition, table extraction.**

I. Introduction

SINGLE Board Computers (SBCs) are widely used in electronic industries, medical instruments, etc. They are low cost and have broad applications from control and measurement to online communication [1], [2]. Typically, SBCs include two parts: one or more main discrete components such as microprocessors, microcontrollers, etc. that target the main goal of the SBC, and second, discrete peripheral components e.g. capacitors, voltage regulators, etc. around the main ones for different functionalities– power, reset, and interface ports to name but a few [3]. It is worth mentioning that discrete components are single electronic components in a simple package which can be soldered to a Printed Circuit Board (PCB) [4].

Finding the desired main components, considering multiple parameters and specifications, is a tedious task for SBC designers as the number of available components is growing at a fast pace. This becomes even more challenging when taking the peripheral components into account. A designer needs to determine various peripheral components, their values and connections. This process is time-consuming, challenging, and will make automated SBC generation a necessity rather than a luxury given the numerous components on an SBC.

In this work, we present a tool, FASCINET, for automatic SBC generation that addresses all the above-mentioned challenges. FASCINET designs customized peripheral circuits and generates the SBC schematic in the industry standard format by giving the main components of the SBC as the input. This design procedure includes all steps of automatically and intelligently determining all the required peripheral components, how they connect to each other and their values e.g. resistance of resistors, capacity of capacitors, model of voltage regulators, etc. FASCINET leverages our novel Hierarchical Deep Neural Network (HDNN) approach to determine the required peripheral components starting with components that could be directly connected to the main components. Then, it hierarchically determines components further away. Next, a separate Deep Neural Network (DNN) model determines the value of each component specified by the HDNN. The generated schematic is passed to a downstream PCB layout generator tool which produces the final PCB of the SBC.

Users have two options for inputting the information of the main components. First, directly inputting the datasheets of the main components. Second, giving the high-level intents and specifications of the main components that they are looking for, and asking FASCINET to find it. For the first case, we propose a novel, open-source[1], fully automated datasheet scrubber to extract the key specifications in the associated datasheet. The datasheet scrubber implements a Convolutional Neural Network (CNN) based approach [5] for category recognition and table extraction on a single platform.

For the second case, FASCINET efficiently searches through its large Commercial Off-the-Shelf Database (COTS DB) in order to find and select an optimal set of components

All authors are with the Department of Electrical and Computer Science, University of Michigan, Ann Arbor, MI, 48109 USA (e-mail: fayazi@umich.edu, zcolter@umich.edu, zinebbe@umich.edu, javadb@umich.edu, ajayi@umich.edu, rdreslin@umich.edu).

[1]Source code for the datasheet scrubber can be downloaded from https://github.com/idea-fasoc/datasheet-scrubber

that meet the user's input requirements. With specifying the target optimization method e.g. power and price, the tool is also able to optimize the SBC in the desired way while satisfying the other user intents such as the specifications of the desired main components. For instance, specifications of a microcontroller include memory capacity, processing power, type of I/O ports, frequency range, throughput, etc. [6]. A user may need two microcontrollers with different specifications as the main components while minimizing their total price. FASCINET searches through the COTS DB and finds the microcontrollers with the closest specifications to each of the desired ones and selects two of them with the minimum total price. This gives a great advantage to FASCINET over a manual design by considering a variety of possibilities, constraints and components which is never possible in a manual design.

FASCINET COTS DB contains meta-data for more than 770,000 records of existing components in more than 50 different categories e.g. microcontroller, voltage regulator, Analog-to-Digital Converter (ADC), etc. Populating such a broad database is tedious and is subjected to errors when done manually. Typically, functionality descriptions of the discrete electronic components are reported in datasheets [7]. Annually, thousands of new components are designed and added to the multitude of existing ones. The datasheet scrubber automatically processes these documents and summarizes their performance specifications which enables creating such a large database. Moreover, for each component, it recognizes the main functionality of each pin, e.g. power, ground, reset, etc. which is an essential step for automatically connecting components.

An exclusive database with more than 2500 open-source SBCs [8], [9] is gathered for training the HDNN model and testing the SBC peripheral circuit generation. To validate the performance, the netlist of more than 400 SBCs designed by FASCINET are compared with the human-designed ones. Our evaluations demonstrate that the SBCs generated by FASCINET are identical to the human-designed ones except for minor differences. Even these minor differences are mainly due to different design approaches, rather than functionality deficiencies.

The contributions of this paper can be summarized as follows:

- Datasheet scrubber: A novel CNN-based approach to categorize datasheets and extract information within their tables.
- Peripheral circuit generator: A novel HDNN-based model that intelligently designs customized peripheral circuits by giving the main components of an SBC.
- We propose FASCINET that leverages the datasheet scrubber, the COTS DB, and the peripheral circuit generator to design the schematic of an SBC in only two minutes.

The rest of the paper is organized as follows. Section II presents the related works. Section III explains the tool in high-level. Section IV describes the datasheet scrubber and approaches for category recognition and table extraction. Section V explains COTS DB management methods. Section VI describes the automated procedure of designing peripheral circuits and generating the SBC schematic. Section VII reports the final evaluation results of each of the aforementioned sections. Finally, the paper is concluded in Section VIII.

II. Related Work

A. Table Extraction

Several methods have studied table identification and extraction. However, many of them focus on tables that are not within PDF format files [10], [11]. On the other hand, Perez-Arriaga *et al.* and Liu *et al.* [12], [13] propose systems for the identification and extraction of tables within PDF files. Perez-Arriaga *et al.* [12] compare the locations of the text-boxes to find columns and rows. Although the method has a decent recall, its false-positive rate is high. Liu *et al.* [13] find tables using grouping texts with similar positions and font size. However, it is only optimized for research papers and its extraction performance is poor especially on general documents. Also, it cannot deal with locked PDFs even though a large amount of datasheets are in locked PDFs.

Recently, many studies have explored table extraction by solely leveraging neural networks [14]–[16]. By utilizing neural networks, table extractors achieve acceptable results in a trade-off with higher computational costs. However, some of the tasks, such as identifying table lines can be performed using conventional computer vision methods. In our approach, instead of forcing neural networks to solve all of the tasks, a combination of both conventional computer vision and machine learning techniques is used. This leads to a neural network that is smaller in size. This is crucial because of the limited amount of labeled training data available for tables.

B. SBC Generation

Several papers and patents have previously described methods for helping users to find the desired components. Birmingham *et al.* [17] propose a method that can help the SBC designer choose various components within a database. Curran *et al.* [18] expand on this and estimate the power, performance, and price of a custom board. The user is prompted for device constraints and the tool recommends components from a library. Likewise, Birmingham and Siewiorek [19] propose an approach which suggests components to designers, then it logically connects those components using templates that describe the purpose of pins. Tools such as these make finding viable components quicker for a designer, however these implementations require a lot of human effort to both continually update their databases and make the components metadata readable by the tools.

Recent works implement more intelligence in suggesting components from their libraries. Anderson *et al.* [20] propose a tool that provides a GUI platform for novice designers. The tool enables users to describe the circuit in a behavioral way. However, it is only compatible with Arduino microcontrollers. When a user searches in library of AutoFritz [21] and Circuito [22] to find a component, the tools also suggest some other components to complete the functionality of the desired component. In AutoFritz, after selecting a component by users, the tool suggests some other components within its database that are usually connected to that component based on its datasheet. It also suggests the possible connections between these components. However, users are supposed to determine at each step whether they want to use any of the suggested components and if so which one. Therefore, there is still a significant amount of human-in-the-loop interaction required. Also, AutoFritz does not get any high-level specifications of the user's goal and limitations as inputs. For instance, users cannot determine the input voltage that they have access to and ask the tool to find the components accordingly or describe the functionality of the components and ask the tool to find the best ones. Moreover, AutoFritz does not support components that do not exist in its database at all. On the other hand, Circuito has fixed pre-defined peripheral circuits for some components that always come with them. However, similar to AutoFritz, it does not get any high-level inputs from users which makes its designs inflexible. Moreover, it does not support components out of its database. The other drawback of Circuito is that there are many limitations in its designs. For instance, it cannot support more than one microcontroller at each design. Our proposed tool gets high-level user specifications as inputs and automatically designs SBCs with a no-human-in-the-loop approach. FASCINET also using ML learns how to use different peripheral components. So, it supports designs that contain components out of its database.

Several works attempt to ease the process of connecting components. Some of them discuss HW/SW co-design approaches to automate generating systems from high-level user intents [23]–[25]. Bachrach *et al.* [25], takes PCB hardware description and application software from the user and generates PCB layout in addition to uploading the software code on it. Lin *et al.* [26] propose a system that supports designers in different levels: High-level Description Language (HDL), electronic modeling (to check the connections, power, etc.) and netlist generation. All the aforementioned works rely on a complete manual hardware description from the user and there is no intelligence by their tool in determining any components or connections. On the other hand, EDASolver [27] after finding components in their libraries, make some connections between them using pin matching. However, this pin mapping is limited to the components in its library and it cannot go beyond that.

The above-mentioned works either help in choosing components or ease connecting them in limited applications while still users have to determine most of the components and connections. Recently Machine Learning (ML) is applied for circuit design as well [28]. By using ML both to keep an up-to-date database of components and to design SBCs, this paper aims to address the shortcomings with the previous works.

III. Platform Architecture Overview

A high-level representation of the FASCINET platform is shown in Fig. 1. Main components are the parts that target the main and overall goal of the SBC. The boxes in green are three inputs that users specify using a config file. Users can choose one of the two ways for inputting the information of the main components: the specifications of them (*path* A in Fig. 1) or directly their datasheets (*path* B in Fig. 1). Beside one of these two, users input the directives that contain three main points:

- The voltage range of the input power supply that the SBC is connected to.
- If the Battery/ USB port is needed.
- The description of the interconnections of the main components to each other.

As it will be discussed in Section VI, the power supply voltage range and the requirement for any Battery/ USB port are two of the HDNN model inputs to determine the required peripheral circuits. There are in general two types of connectivities for a main component. It is either connected to another main component or a peripheral component. In the first case scenario, the description of interconnections of the main components which is provided by the user is used for automatically connecting them. As an example, a user may specify the clock pin of a main component is connected to the clock pin of another main component. As it will be explained in Section IV-C, the datasheet scrubber is able to recognize pins functionality and find the exact pin, e.g. clock in this example. The procedure of automatic connection of the main components to peripheral ones as well as peripheral components to each other will be described in Section VI.

If users decide to input the specifications of the main components (*path* A in Fig. 1), they also would be able to determine some constraints. The total SBC main components power consumption at nominal voltage and their total price budget are the constraints that the SBC can be optimized for. The tool outputs the closest components that it could find within its COTS DB regarding these overall constraints. If the tool finds more than one component that meet the required specifications given by the user, it picks the one with the lower price or power consumption based on the desired optimization approach. The output of the COTS DB is a complete summary of the specifications of the found main components and pins functional information which were parsed previously from the associated datasheet by the scrubber tool.

In the case of directly inputting the information of the main components by users (*path* B in Fig. 1), the datasheet scrubber parses the datasheets using COTS DB training data and passes the summarized overall and pin functional meta-information to the peripheral circuit generator.

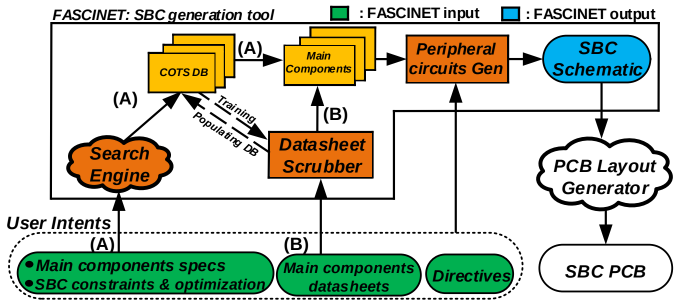


Fig. 1. High-level platform architecture.

The summarized information of the main components includes their categories, specifications, and other information required for proper functionality. For instance, this summarized information may be: an 8-bit ADC with an input voltage range between $2-5V$. Using this meta-information in addition to the directives, the peripheral circuit generator creates the SBC schematic in the industry standard format as an output which contains all the main and peripheral components, and their connections. This schematic can be passed to a downstream PCB layout generator tool.

IV. Datasheet Scrubber

A. Category Recognizer

The first step to automatically scrub datasheets is to recognize the category of components, e.g. ADC, Phase Locked Loop (PLL), etc. Although the category of some datasheets found in some websites may have been labeled, there are many datasheets without labeling that need manual category recognition and separation which wastes much time and energy and is subjected to errors. Moreover, these labels may not be precise enough. For instance, a category such as PLL might need to be broken into two subcategories e.g. analog PLL and Delay Locked Loop (DLL). Using the datasheet scrubber, this procedure can be done automatically with no-human-in-the-loop which saves a lot of time and energy.

Category recognition is a crucial task since the scrubber should look for different specifications based on the category of the components. In order to effectively and efficiently identify the the correct category, a customized CNN-based model is used which identifies between 51 common categories. It is worth mentioning that some existing approaches such as searching the categories name within the datasheet and selecting the most frequent one might be simpler, but are subjected to a lot of errors. For instance, the datasheet of a microcontroller may talk more about the memory and contains more "memory" keywords rather than "microcontroller". Another not accurate enough approach is searching the categories name within the title. The problem is that mentioning the category in the title is not a mandate and there are many datasheets without having the category mentioned in their titles. Moreover, automating the procedure of realizing the title for the tool would be complicated enough despite it being trivial for humans. Also, the title may contain more than one category. Furthermore, Support Vector Machines (SVMs) have long been used for the text classification and they have achieved reasonable success depending on the database. The general procedure, when using an SVM, is to identify the quantity of each word in a document, then train the SVM using those quantities as inputs [29]. This approach would not produce accurate results for our database due to the similarity between different categories. For instance, when differentiating between an ADC and a Digital to Analog Converter (DAC), the order of the words is very important. This issue can be somewhat resolved by searching for phrases instead of words, but choosing the proper phrases to use is a non-trivial and labor-intensive process. As a result, our customized CNN-based approach is designed to be as simple as possible while having an acceptable accuracy.

The customized CNN-based model, instead of examining the entire document, analyzes the first 256 words. There are several factors that determine the ideal amount of words to examine per document. Looking at more words allows more information to be evaluated by the network. However, this leads to a wider network, more memory, and a higher computational power. Analyzing hundreds of datasheets shows that a vast majority of them front-load their important information with a short description. This description most often varies between 200 to 250 words in length. With this information, the size of the network has been optimized to take in all of this concentrated data in a vast majority of cases.

The first 256 words are converted to vectors of 300 numbers with the GloVe word embeddings [30]. This step allows the network to understand how words relate to one another and crucially which words are synonyms. A conventional CNN approach would then stack many convolutional layers that are followed by several dense layers. The stacked convolutional layers allow relationships to be discovered between distant words. However, this method has several downsides. Making a robust model with this approach requires a massive amount of weights, increasing the training time and making the model less portable. Another issue is the vanishing gradient problem that arises from stacking many layers [31].

Our customized network has an innovative design that identifies relations between distinct words without the need for a significant number of stacked layers. In this model, there are several branches that compute unique relations between the words. The simplest branches start with filters that are single word in length, so each filter has a size of 1×300. This is in practice a 1D convolution because the size of the filter is equivalent to feature map size in one dimension. The output of this operation is a two-dimensional array where one dimension still represents the location of the words, but the other represents each filter. This array shows how closely the word represented by a particular filter matches the meaning of the input word. The next step is unconventional where the two axes are permuted before the data is passed into another convolutional layer.

The size of the filter in the next layer, like the previous one, is equivalent to the length of the feature map in one dimension, making its functionality similar to a 1D convolution. However, because the data is permuted, this filter travels across the dimension created by the number of filters in the previous step. This means that it has access to data created by every input word simultaneously. Therefore, it can identify the relations between different words that are an arbitrary distance away. The length of the filter determines the amount of word-meanings that can be compared at each stride. It should be noted that the objective function for this CNN is cross entropy.

This approach does have some downsides compared to a conventional Deep Convolutional Neural Network (DCNN). Because the second convolutional layer looks for words in specific places, it might not be general enough. To rectify this issue, many different branches are used to cover any potential weaknesses. These branches have different filter lengths for the first convolutional layer. This causes phrases to be identified instead of words, allowing the second layer to identify different

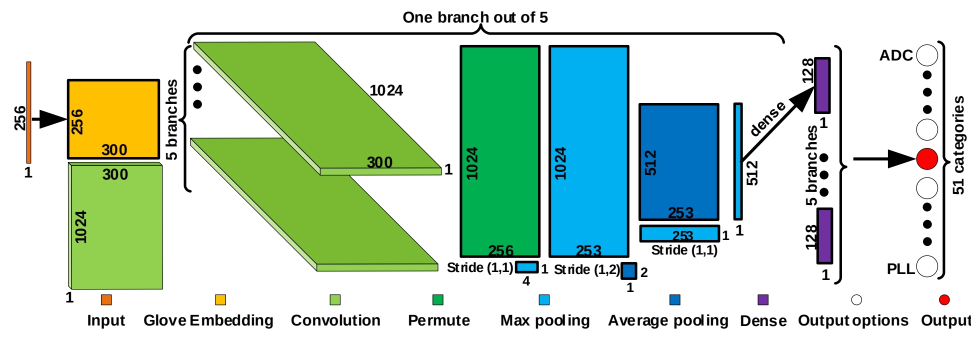


Fig. 2. The CNN model of the category recognizer. The input is the first 256 words of datasheets and output is one of the 51 categories.

phrases that are an arbitrary distance away. Additionally, some of these branches have maxpooling layers between the two convolutional layers, helping regain generality.

The CNN model has been depicted in Fig. 2. There are in total 5 different branches. Each branch has a specific length of the first filter that are chosen based on the testing results as it will be more discussed in Section VII-A1. In the final step, all of the branches are flattened and then concatenated. The combined network ends with nodes that represent all the different possible categories. To train this network, we use the 770,000 presorted datasheets in the COTS DB.

B. Table Extractor

A majority of relevant specifications and pin information of a datasheet are found within its various tables. However, an agreed upon format for the layout of tables in datasheets does not exist. Although there have been some studies on automation of table extraction as it will be discussed in Section II, most of them are either not able to extract all PDF files, or they do not have enough accuracy, or they are computationally expensive. However, our method combines computer vision techniques and CNN models that allows CNNs to just focus on complex problems while computer vision methods take care of the rest. This approach leads to a high accuracy with a low computational cost.

Our customized table extractor starts by converting the datasheets into a series of images. A CNN then goes through these images and finds the locations of the tables [32]. The CNN model for the table position identification is shown in Fig. 3. Afterwards, computer vision methods are used to identify the high-level layout of the tables that helps identify and separate cells: First, the locations of high contrast lines within the tables are identified. Additionally, the locations of text within the tables are found by locating high contrast sections that do not belong to the previously found lines.

The identified high-contrast lines are useful for separating cells. However, many tables do not have lines separating every individual cell. An example of this case is shown in Fig. 4 (a). In this case, the positions of the data can be used instead. With the locations of the data known, the tool finds these invisible cell separator lines. For this purpose, the tool draws straight lines that do not intersect the data all over the table. Fig. 4 (b) shows such vertical lines in green. For being more clear, the horizontal lines are not shown. Assuming l_{vm} and l_{hm} denote the maximum length of vertical and horizontal drawn lines respectively, the tool only keeps vertical and horizontal lines with l_{vm} and l_{hm} lengths respectively, and removes the rest. Finally, the adjacent remained lines are merged as it is depicted in Fig. 4 (c).

This process alone is sufficient to properly split a table with consistent rows and columns across the entire table. However, more complex tables which contain cells that do not conform to the general table's layout will have incorrectly split and merged cells. For instance, a single cell may be larger than its adjacent cells and may span two columns. To fix these issues, the cells are sent into a CNN [32] with the purpose of finding which cells should be merged together and which cells should be split. The fixed cells are then individually sent into the open-source Optical Character Recognition (OCR) software Tesseract [33] to obtain the cell's data. The data and the previously identified locations of the cells are used to recreate the table in a machine readable form.

When the table extractor is used for extracting the specifications of main components in both *path* A and B in Fig. 1, there are some common specifications that FASCINET always searches for e.g. power, input voltage range, etc. However, the rest are more dependent on users that which specifications are more important for them and the category of the desired component.

C. Pin Functionality Extractor

After finding the location of the pin description table within a datasheet using the table extractor, the data needs to be interpreted. It is important to know the purpose of each pin for automatically connecting components. The pin functionality extractor identifies if a pin is a ground, power, clock, reset, General-Purpose Input Output (GPIO), etc. For this purpose, a CNN looks at both the pin name and the description.

To learn from the pin name, the CNN takes in each of the individual characters as one-hot vectors. This is done so that patterns within similar pin names can be learned. For instance, "VDD" almost always indicates a pin that should be connected to the power supply. This is true regardless of any additional characters that are in the pin name, so "DVDD" and "VDD0" can both be classified as a power pin.

Additionally, a glove matrix which has 4 layers of 1D convolution followed by a CNN, with cross entropy as the

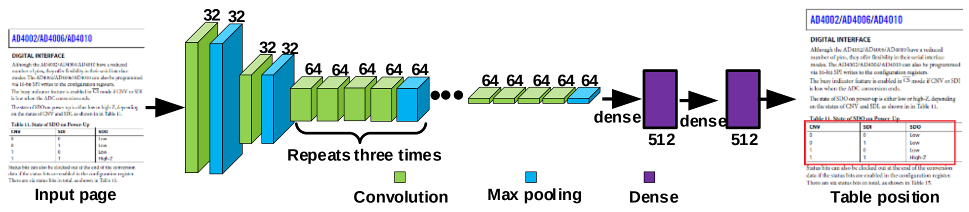
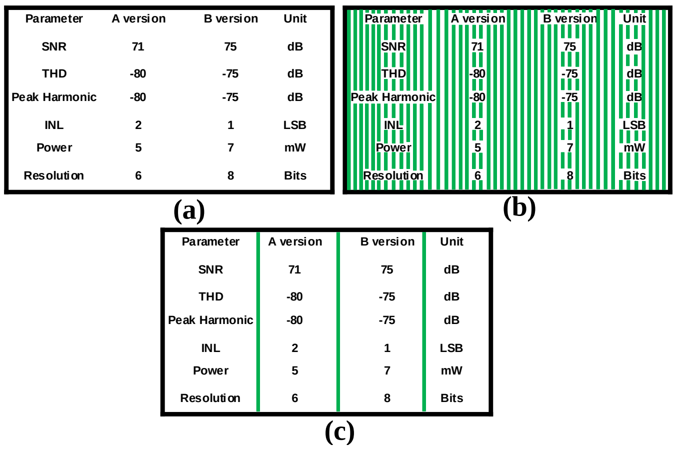


Fig. 3. The CNN model of the table position identification. The input is the datasheet page and output is the position of the table within it.



(a)

Parameter	A version	B version	Unit
SNR	71	75	dB
THD	-80	-75	dB
Peak Harmonic	-80	-75	dB
INL	2	1	LSB
Power	5	7	mW
Resolution	6	8	Bits

(b)

Parameter	A version	B version	Unit
SNR	71	75	dB
THD	-80	-75	dB
Peak Harmonic	-80	-75	dB
INL	2	1	LSB
Power	5	7	mW
Resolution	6	8	Bits

(c)

Parameter	A version	B version	Unit
SNR	71	75	dB
THD	-80	-75	dB
Peak Harmonic	-80	-75	dB
INL	2	1	LSB
Power	5	7	mW
Resolution	6	8	Bits

Fig. 4. Finding invisible cell separator lines. (a) An example of table without cell separator lines. (b) The tool draws straight lines that do not intersect the data all over the table. (c) The final result after removing shortened lines and merging the adjacent ones. For being more clear the horizontal lines have not been shown.

objective function, is used to learn from the description of the pins at a word level. The CNN models for both learning from pins name and descriptions have two hidden layers of 1D convolution. These two networks are concatenated together in order to achieve accurate results.

V. COTS Database

In order to find the desired main components based on the user specifications, a database has been configured to allow for immediate access to the collected data. The COTS DB contains more than 770,000 component records from different sources [8], [9], [34]. Because the category of each component has unique specifications, a MongoDB platform is adopted for its flexible data-scheme. MongoDB is a document-oriented NoSQL database that offers high performance, scalable, flexible data-schema [35]. From the performance perspective, MongoDB has been proven to be effective for storing a huge amount of data [36] and it outperforms MySQL in read and write tests [37].

Optimizing the search performance within the COTS DB is crucial to reduce the latency. When a query operation is performed, MongoDB performs a collection scan, meaning it scans every record in the collection to select the ones that match the query statement. Multiple methods such as indexing, projections, and aggregation pipelines [38] are implemented to optimize the COTS DB.

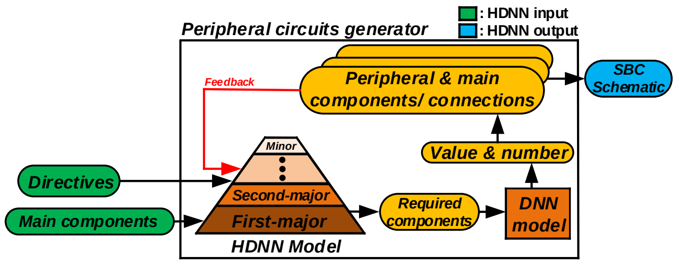


Fig. 5. Peripheral circuit generator flow using the HDNN model.

Indexes are special data structures that store a small portion of the collection's dataset in an easy-to-traverse form. This optimizes the execution of queries [39]. The indexing is applied to the price and power of the components. The projections method decreases the query time by limiting the number of either searched or returned fields. Before the projection, the initial number of searched fields was 28 while they have been shrunk to 5 fields based on the most commonly performed queries.

Using the COTS DB, FASCINET is also able to minimize both the total power consumption and price of the main components with regarding to the specifications of the main components. Moreover, FASCINET has the feature of prioritizing the specs which enables users to specify the more important specs for them. In this regard, even if the queried component do not exist within the COTS DB, FASCINET finds the most close one to it.

VI. Peripheral Circuit Generator

The main components specify the main intention of SBCs and target their main goals. Using the summarized information of the main components e.g. categories, specifications, etc. (that is either extracted by the datasheet scrubber or found within the COTS DB based on the user input), in addition to the directives (that are inputted by users directly), the peripheral circuit generator designs the peripheral circuits around the main components. It then outputs the SBC schematic in the industry standard format. For this purpose, the peripheral circuit generator uses our novel Hierarchical Deep Neural Network (HDNN) model.

Fig. 5 depicts the peripheral circuit generation flow. Given the main components and directives by the user, the HDNN model determines the required peripheral components that could be directly connected to the main components ("first-major"), then it hierarchically and successively determines the

TABLE I
SUMMARIZED HDNN'S INPUTS, THEIR SOURCES AND THE NUMBER OF COMPONENTS THAT CAN BE DETERMINED AT EACH PERIPHERAL SECTION.

Peripheral section	Inputs							Determinable peripheral components #
	Power pins #*	Input voltage range	Battery/ USB port is needed	LED pins #	Components category	Reset pins #	Clock/ oscillator pins #	
Power supply	M, P**	M, P, D	D	-	-	-	-	57
LED	-	-	-	M, P	M, P	-	-	8
Reset	-	-	-	-	M, P	M, P	-	7
External interfaces	-	-	D	-	M, P	-	-	10
Clock	-	-	-	-	-	-	M, P	3

* #: Number.
** Inputs sources: Main components (M), Directives (D), Peripheral components (P).

other required components that are further away ("second-major", ..., "minor"). Then, a DNN model determines the values e.g. resistance of resistors, etc. of these peripheral components that are determined by the HDNN model. As an example, the HDNN model determines if a decoupling capacitor is required to be connected to the main component, and the DNN model after that determines its value i.e. the capacity. At the end, unconnected pins would be connected to the correct signal based on the extracted pin functional information by the datasheet scrubber and inputted directives. Finally, the SBC schematic is generated.

The HDNN's inputs come from three different sources. Main components and directives that are fixed at runtime and specified by the user. The third input source is the previously generated peripheral components which is updated during runtime. As the feedback loop is shown in Fig. 5, the HDNN model runs multiple iterations to design all peripheral components. It is necessary to update the inputs based on the previously generated components because of the dependencies between them. For example, adding an LED increases the total required power. To account for this, the updated circuit is sent back into the model so that it can determine the peripheral power supply section accurately. Table I summarises the HDNN's inputs and their sources as well as the number of peripheral components that can be determined by the HDNN for each peripheral section e.g. power supply, LED, reset, etc. In total, there are 7 different inputs to the HDNN model and the HDNN is able to determine 85 different peripheral components. It should be noted that "components category" is considered as one input set which contains the category of all peripheral and main components. Also, "battery/ USB port is needed" is divided into two inputs. Each of the peripheral sections contains the components that together have a specific functionality. For instance, power supply section provides constant, stable voltage for whole the SBC and may include some capacitors, voltage regulators, diodes, etc. that are determined by the HDNN model.

As it is shown in Fig. 6, the HDNN model uses a custom made binary classifier in Keras [40] with 2 Fully Connected (FC) hidden layers. The hidden layers have 32 nodes each with a nonlinear RELU activation function, while the output layer has a nonlinear sigmoid activation function and cross

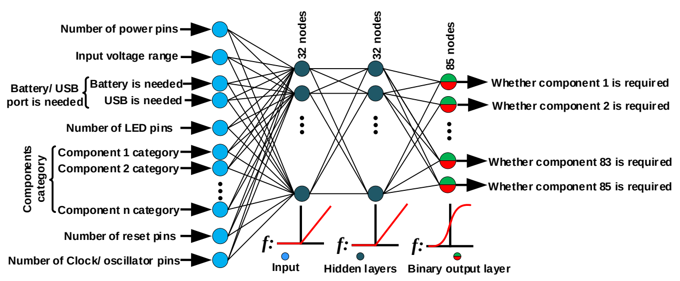


Fig. 6. HDNN model: A custom made binary classifier in Keras [40] with 2 Fully Connected (FC) hidden layers. The hidden layers have 32 nodes each with a nonlinear RELU activation function, while the output layer has a nonlinear sigmoid activation function. The inputs with more details are described in Table I.

entropy as the objective function to determine the required peripheral components. As it is shown in Fig. 5, it begins by determining the required components at the first level of the hierarchy, first-major, that could be directly connected to the main components in each peripheral section e.g. power supply, LED, reset, etc. First-major components may be surrounded by other components, second-major, and so on. The components at the last level of hierarchy, minor, are the ones that already all of the components that they are connected to have been determined by the model and usually one of their pins are connected to ground e.g. decoupling capacitors.

In the HDNN model, the binary classifier at each step determines whether a peripheral component that is observed in the training set is required for the SBC. Those 85 different peripheral components that are determinable by the HDNN model (see Table I), regardless of their values, are different components with different functionalities that are connected to unique nets and have been observed at least in one of the training sample designs [8], [9]. For example, the LED that is connected to a specific pin of a main component and the one that is connected to the power signal are considered as two different components. As another example, voltage regulators are part of the power supply peripheral section that usually are connected directly to the main components. In our training set, there are four different voltage regulator components which means some SBCs have been observed that each has one or more of these voltage regulators. For instance, one voltage regulator may have 3 pins such as "LM117" [41] that is used usually when the input voltage is less than $40V$, and the other may have 5 pins such as "AP2112" [42] which is leveraged if the input voltage is between $2-6V$. So, the HDNN determines whether the SBC requires any of these voltage regulators.

Automatically connecting the peripheral components to each other and the main ones through the corresponding pins are implied by determining the required peripheral components by the HDNN model. The pin information extracted by the datasheet scrubber is used for this purpose. In other words, during training, the HDNN model not only learns which peripheral component are required to be connected to the main ones, but it also learns through which pins they should be connected to. For example, determining requirement of a voltage regulator around the main component by the HDNN model means that the output pin of the voltage regulator is connected to the power pin of the main component. So, determining which pin of the main component works as power is the last piece of puzzle for automatically connecting them that pin functionality extractor part of the datasheet scrubber does that. Finally, FASCINET automatically connects all ground pins to ground, all power pins to the appropriate power supply value using their datasheets extracted information, and main components to each other based on the input directives of users. FASCINET also connects all the main components pins to headers. This allows users always have access to inputs and outputs of the main components directly.

The step by step hierarchically design procedure of a sample SBC has been depicted in Fig. 7. After passing the information of the main components, the peripheral components that are directly connected to the main components, i.e. first-major are determined. A similar design procedure that was described for determining the voltage regulators are happening for this and the next steps. As an example, in Fig. 7 (c), for determining which power signals should be connected to the voltage regulator ("AP2112"), there are a lot of options that among them "V_USB" and "V_BAT" are selected.

After determining the required peripheral components, a multi-label DNN classifier determines the values of these peripheral components e.g. resistance of resistors, capacity of capacitors, etc. Similar to the HDNN procedure, for each peripheral component, the value is determined among all values that have been observed for that component within the training set. For instance, for a decoupling cap, the DNN model determines which one of the values of $47uF$, $10uF$, $1uF$, $100nF$, etc. that are observed in the training should be chosen. Using a multi-label classifier is better than a normal regression for this purpose as the values that may be obtained by the regression model may not even exist e.g. there is no $5.34\mu F$ capacitor. Occasionally, multiple components of the same category are connected in parallel [43], e.g. capacitors that are connected between supply voltage and ground as it is depicted in Fig. 7 (c). In these cases, the DNN will also determine the number of each component. It should be noted that sometimes the HDNN model determines the value of components as well in the cases when their values affect the rest of the design. Voltage regulator is an example of this since specific components are connected to each of them based on their models.

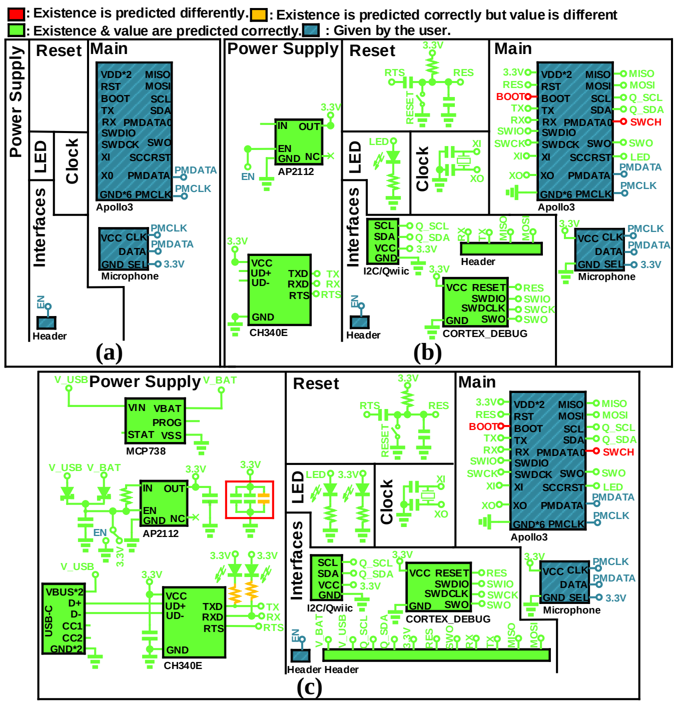

🟥: Existence is predicted differently.
🟨: Existence is predicted correctly but value is different
🟩: Existence & value are predicted correctly.
▨: Given by the user.
Power Supply
Reset
Main
LED
Clock
Interfaces
Apollo3
Microphone
Header
AP2112
CH340E
I2C/Qwiic
CORTEX_DEBUG
MCP738
USB-C
(a)
(b)
(c)

Fig. 7. Steps of determining the required peripheral components by the HDNN model: (a) The main components are inputted. (b) The peripheral components that are directly connected to the main components, i.e. first-major, are determined. (c) The peripheral components around the first-major components, i.e. second-major, are determined. The next step which is the complete SBC after determining all the required peripheral components has been shown in Fig. 15. An example of parallel capacitors that their number is determined by the DNN model is shown with a red box.

The training set for both the HDNN and DNN models contains more than 2500 open-source SBCs [8], [9]. A hierarchical approach is also used for training the HDNN model [44]. A unique training set is provided for determining peripheral components around each component. There is one training set for determining first-major components. Each of these first-major components has its own specific training set for determining second-major components around them and so on. For instance, two different training sets are provided for determining components around "AP2112" [42] and "LM117" [41] voltage regulators that are trained separately on each. This idea of hierarchical training leads to an optimized result for each as well as reducing the complexity of the network [44].

There are other SBCs which contain peripheral components that are not included in any designs in the training set (e.g. transformers). However, FASCINET would be able to support such components easily if a proper number of SBC designs that include this new component is added to its training set. Also, it is probable that more directives from the user are needed while the rest of the HDNN procedure is the same. The HDNN model is optimized to support current 85 components with the minimum number of training data and directives while maintaining a high accuracy. While the whole HDNN procedure would be the same, adding new components is with cost of having more training data and directives from the user. This leads to have a more complicated model which not only

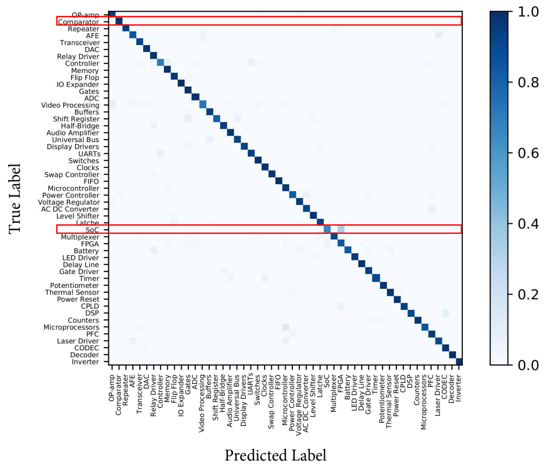


Fig. 8. The confusion matrix of the category recognizer when the first 256 words are given as inputs. Red rectangles show that, generally, common categories (e.g. Comparator) with a larger training set have a higher accuracy in comparison to the niche ones (e.g. SoC).

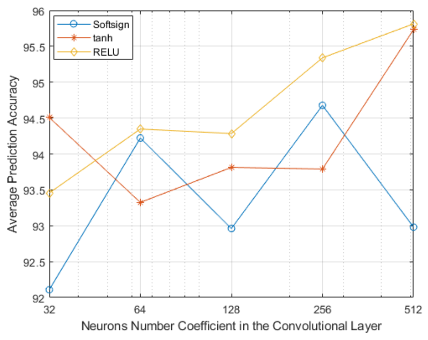


Fig. 9. Average prediction accuracy comparison of the category recognizer with different hyperparameters used for training. As it was mentioned in Section IV-A, there are 5 branches with different neuron numbers that at the end are flattened and concatenated. The base neurons numbers in the convolutional layer of these branches are $\{1, 2, 2, 2, 4\}$ that the coefficient in the X-axis is multiplied to them. For instance, neuron number coefficient of 32 leads to a network which its branches have $\{32, 64, 64, 64, 128\}$ neurons in their convolutional layer. The figure shows different activation functions applied to the convolutional layer. The coefficient of 256, i.e. $\{256, 512, 512, 512, 1024\}$ neurons, with the RELU activation function is the simplest architecture that gives a high prediction accuracy of 95.3%. The first 256 words are given as the input.

TABLE II
AVERAGE ACCURACY OF THE CATEGORY RECOGNIZER AND THE TRAINING TIME FOR DIFFERENT NUMBER OF INPUT WORDS. THE TRAINING TIMES ARE NORMALIZED TO THE TRAINING TIME OF 512.

Input words #*	128	256	360	480	512
Average prediction accuracy	91.73%	95.3%	95.2%	94%	95.7%
Normalized training time	0.28	0.53	0.72	0.94	1

* #: Number.

TABLE III
TABLE EXTRACTION PRECISION AND RECALL COMPARISON.

Method	Precision	Recall	F1-Score
CNN-based approach	96.3%	93.5%	94.88%
Tabula-based approach	82.8%	83.0%	82.9%

needs more data to be trained, but also needs more runtime.

VII. Evaluation

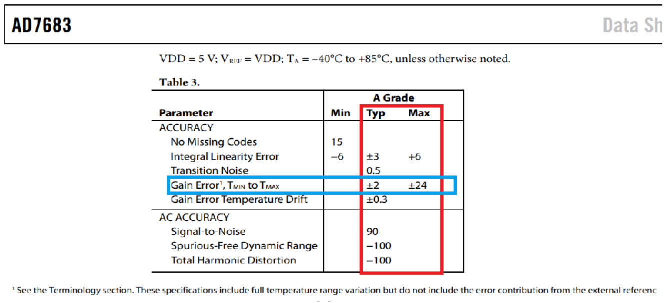
A. Datasheet Scrubber

1) Category Recognizer: To identify the accuracy of the model, a stratified k-fold with 5 splits is used with the labeled categories [8], [9], [34] as the ground-truth. The first 256 words are given as inputs. Using this method, the model has an average accuracy of 95.3% on the test set. Fig. 8 shows a confusion matrix of the predicted categories. In order to compare the results, the Bidirectional Encoder Representations from Transformers (BERT) text classification [45] is also analyzed on the same training and test sets. For this purpose, “Bert For Sequence Classification” model is used and tested with two test cases. For one test case scenario, the first 128 words are given as inputs with 256 tokens while the other case uses the first 256 words with 360 tokens. They give 92.7% and 92.9% average prediction accuracy respectively, which are less than the proposed CNN method.

Table II summarizes the model prediction average accuracy and the training time for different number of input words. As you can see, even though the accuracy of 512 words is 0.4% better than the 256 one’s, its training time is 1.88× more than the 256 one’s. This shows almost no improvement is gained while the cost of computation time has gotten roughly double if the first 512 words are given as inputs instead of the first 256 words. Except for the 512 model, the accuracy of the 256 model is better than others. Fig. 9 shows the average prediction accuracy with different hyperparameters.

2) Table Extractor: Over 1000 tables within various datasheets are used to train the model. The line identification from Tablext [32] is used to pre-process the tables by splitting them into cells. Adjacent cells are merged and used to train the CNN model. Merged cells that should be merged according to the ground truth are labeled as such. For validation of the table extractor, a test set that contains 200 random PDF datasheets, not used in any training, is procured. The correct specifications are manually searched for in each table within the datasheets. These manually found specifications are compared to the extracted results from the table extractor in order to determine the precision and recall. In the formulas below, *TP* denotes true-positives (correctly extracted cells) and *FP* denotes false-positives (non-existent cells that are misidentified by the extraction approach). Also, *FN* represents false-negatives (existing cells in the tables that are not accounted for or misidentified by the extraction tool). Precision, recall, and F1-Score can be calculated by $\frac{TP}{TP+FP}$, $\frac{TP}{TP+FN}$, and $\frac{2*\text{Precision}*\text{Recall}}{\text{Precision}+\text{Recall}}$ respectively.

Our customized CNN-based approach is compared with an open-source table extractor tool, Tabula [46], and the results are summarized in Table III. Our CNN-based approach achieves higher precision and recall, despite not having access to any text data, unlike Tabula. Fig. 10 shows an example of the CNN-based table extractor and Tabula’s final output in



AD7683 Data Sh

VDD = 5 V; V$_{REF}$ = VDD; T$_A$ = −40°C to +85°C, unless otherwise noted.

Table 3.

Parameter	Min	A Grade Typ	Max
ACCURACY			
No Missing Codes	15		
Integral Linearity Error	−6	±3	+6
Transition Noise		0.5	
Gain Error¹, T$_{MIN}$ to T$_{MAX}$		±2	±24
Gain Error Temperature Drift		±0.3	
AC ACCURACY			
Signal-to-Noise		90	
Spurious-Free Dynamic Range		−100	
Total Harmonic Distortion		−100	

¹ See the Terminology section. These specifications include full temperature range variation but do not include the error contribution from the external referenc

(a)

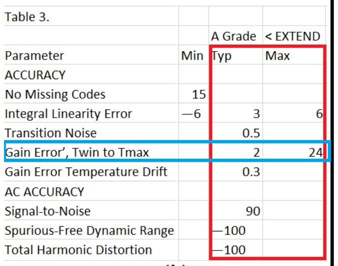
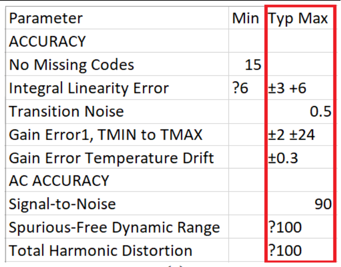


Table 3.			
		A Grade	< EXTEND
Parameter	Min	Typ	Max
ACCURACY			
No Missing Codes	15		
Integral Linearity Error	—6	3	6
Transition Noise		0.5	
Gain Error', Twin to Tmax		2	24
Gain Error Temperature Drift		0.3	
AC ACCURACY			
Signal-to-Noise		90	
Spurious-Free Dynamic Range		—100	
Total Harmonic Distortion		—100	

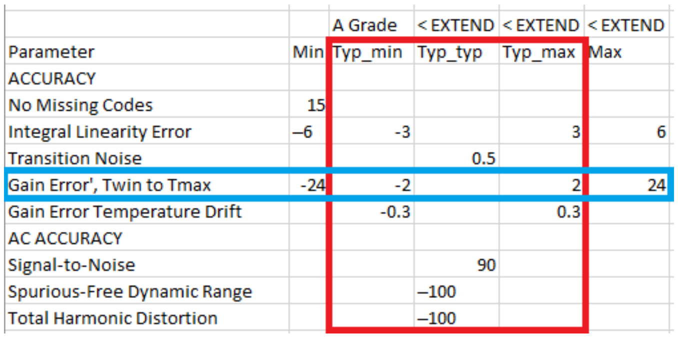
(b)



Parameter	Min	Typ Max
ACCURACY		
No Missing Codes	15	
Integral Linearity Error	?6	±3 +6
Transition Noise		0.5
Gain Error1, TMIN to TMAX		±2 ±24
Gain Error Temperature Drift		±0.3
AC ACCURACY		
Signal-to-Noise		90
Spurious-Free Dynamic Range		?100
Total Harmonic Distortion		?100

(c)

Fig. 10. (a) A text-based PDF page in the testbench with potentially problematic columns highlighted with boxes (red). The final output of (b) CNN-based approach, (c) Tabula-based approach in CSV format given (a) as the input PDF file. Vertical boxes (red) are showing the correct handling of problematic cells. Tabula has merged "Typ" & "max" columns and shows "-" character as "?" while the CNN-based approach correctly has handled both problems. On the other hand, the CNN-based approach cannot show "±" since Tesseract appears not to be recognizing this character. The horizontal box (blue) shows when ± is written for just one of Max and Min columns and the other is blank.



		A Grade	< EXTEND	< EXTEND	< EXTEND
Parameter	Min	Typ_min	Typ_typ	Typ_max	Max
ACCURACY					
No Missing Codes	15				
Integral Linearity Error	–6	-3		3	6
Transition Noise			0.5		
Gain Error', Twin to Tmax	-24	-2		2	24
Gain Error Temperature Drift		-0.3		0.3	
AC ACCURACY					
Signal-to-Noise			90		
Spurious-Free Dynamic Range			–100		
Total Harmonic Distortion			–100		

Fig. 11. The output table after applying the post-processing on Fig. 10 (b) to be used in the SBC generation flow. Vertical boxes (red) show the three new sub columns that are replaced with "Typ" column to rectify the not-showing the ± character. "Typ_min" and "Typ_max" contain negative and positive value of the specs with ±. "Typ_typ" is the third sub column that is used for the typical value of the other specs that do not use ±. The horizontal box (blue) shows after the post processing both "Max" and "Min" columns are filled for the specs that ± is written for just one of them and the other is blank.

Comma-Separated Values (CSV) format.

As it is depicted in Fig. 10, even though our CNN-based approach recognizes + and − signs, it is not able to show "±" since Tesseract appears not to be recognizing this character. The tool performs a post-processing for making the table appropriate to be used in the SBC generation flow. Our analysis shows that ± appears only for certain specifications and it has two meanings in general: 1) showing a range (e.g. Integral Linearity Error [47]); 2) when both + and − values are applicable (e.g. leakage current [48] may be reported with ± which means it is either from ground to the component or vice versa). After recognizing such certain specs that use ±, in the post-processing, the "Typ" column is divided into three sub columns: "Typ_min", "Typ_typ", and "Typ_max". The table extractor reports the absolute value, so + and − signs are added for Typ_max and Typ_min respectively. For "Max" and "Min" columns of such specs, if just one of them has a value (e.g. the horizontal blue box in Fig. 10), in the post processing, + and − signs are added to the absolute value for the "Max" and "Min" columns respectively. The result of applying post-processing on Fig. 10 (b) is shown in Fig. 11.

3) Pin Functional Extractor: To test the pin extraction network, the pins from 2000 datasheets are collected and split 80, 20 into training and testing datasets. The network achieves 96.5% accuracy on the testing dataset, proving its effectiveness.

TABLE IV
COMPARISON OF THE SEARCH OPTIMIZATION METHODS ON THE COTS DB.

Method	Time
Without indexing	0.037s
With indexing	0.006s
With indexing and projection	0.005s
With indexing and aggregation pipelines	0.002s
Without indexing and aggregation pipelines	0.001s

B. COTS DB

Table IV compares the search time for different methods. A search is performed on 28 fields of a collection of more than 770,000 datasheets. Implementing indexing optimizes the query search time by approximately 6×, while the aggregation pipeline method achieves a 37× speedup in comparison to a normal search.

C. Peripheral Circuit Generator

To test the SBC generator tool, 400 SBCs are used that are independent from the training set. For this purpose, the main components of 400 human-designed schematics are given to the FASCINET as input. This allows to fairly compare the generated SBCs by FASCINET with the human-designed ones as ground truth. All experiments are performed on a server with an NVIDIA TITAN V GPU. For giving the main components information, both directly inputting their datasheets, and describing their specifications and searching within the COTS DB have been tested. In the case of searching the specifications of main components, the ability of FASCINET to optimize the total price, power, or meeting the total price/power budget constraints are also tested in addition to satisfying the desired specifications. In such a scenario, FASCINET is limited to components that are within its COTS DB. This means if the COTS DB does not contain any component with the exact desired specs, the search would not be successful. However, in this regard, FASCINET finds the most close components within its COTS DB to the requested ones using the feature of prioritizing the specs.

TABLE V
MAIN COMPONENTS SEARCH RESULTS WITHIN THE COTS DB. MAIN COMPONENTS THAT ARE USED IN 400 HUMAN-DESIGNED SBCS ARE SET AS THE TARGET OF THESE SEARCHES.

	Meet specs	Meet specs & equal power	Meet specs & equal price	Meet specs & better power	Meet specs & better price	Meet specs & meet power/price budget
Total search #*	400	150	150	150	150	100
Successful search #	398	141	135	7	15	100

* #: Number.

TABLE VI
DRC RESULTS FOR THE PROPOSED PERIPHERAL CIRCUIT GENERATOR; THE TEST SET INCLUDES 400 SBCS.

Total components #*	Total main components #	Total pins #	Total connections #	Floating (open) ports #	Short-circuit #	Free DRC error design #	Design number with DRC error
16,600	850	50,000	41,500	750	15	278	122

* #: Number.

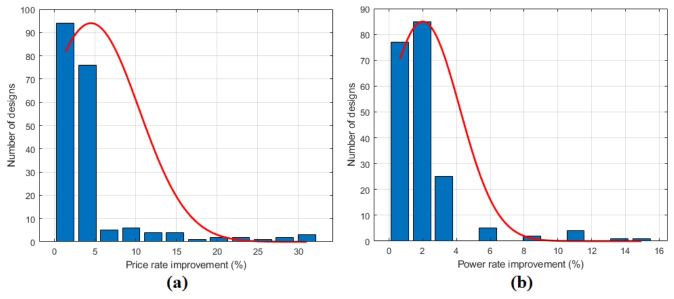


Fig. 12. The distribution of improvement rate of (a) price and (b) power of found components by the search engine over the human-designed ones. Improvement here means less for both power and price.

TABLE VII
PERIPHERAL CIRCUIT GENERATOR ACCURACY IN COMPARISON WITH THE HUMAN-DESIGNED SBCS; THE TEST SET INCLUDES 400 SBCS.

Identical connections	Precision	Recall	F1-Score
87%	97.9%	96%	96.9%

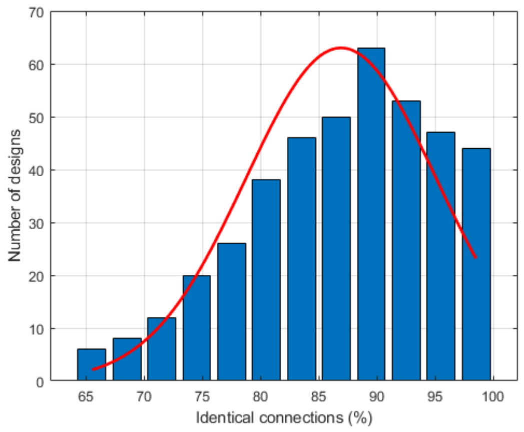


Fig. 13. The distribution of identical connections over the compared designs.

For each minimum total price and minimum total power consumption of main components, 150 searches are performed to find the main components that are used in human-designed SBCs. For the 100 rest, the total power and price of the main components in each of the human-designed SBCs are set as the budget constraints. The found components within the COTS DB are compared manually, both in terms of meeting the specifications and price/power constraints, with the target main components that are used in human designs. The results are summarize in Table V. For instance, if the found main components meet the performance requirement, i.e. having the same specifications as the ground-truth main components in the human-designed ones, while having equal and less total power/price, it is counted as successfully “meet performance & equal power/price” and “meet performance & better power/price” respectively. As it is shown in Table V, there are only two unsuccessful searches because, for these two, the COTS DB does not contain any components with the same specifications with the requested ones. However, using prioritizing feature of FASCINET, the most close components within the database is outputted. In order to fairly compare the schematic generation of FASCINET with the human-designed ones, the exact main components that are used in the human-designed schematic are given to FASCINET regardless of the COTS DB output. The distribution of improvement rate of price and power of found components by the search engine over the human-designed ones are depicted in Fig. 12.

In addition to testing the satisfaction of main components constrains, all of the generated schematics are evaluated to check whether they are valid. For this purpose, a Design Rule Check (DRC) is run to check any floating (open) port or any short-circuit connections. As the results are summarized in Table VI, we can see that less than 1.5% of ports of the components are float. Also, the percent of short-circuit connections are negligible. Even this small error is due to misunderstanding the functionality of the pins, not the SBC peripheral generator itself. Because, as it is discussed in Section VI, the HDNN model selects the components around each component among the ones that are connected to that component in the training set. Therefore, it is probable that the HDNN model does not realize where should it connect the pins to and leaves them float. But, it is impossible to connect them to wrong pins (short-circuit) as there is no sample like that in the training set.

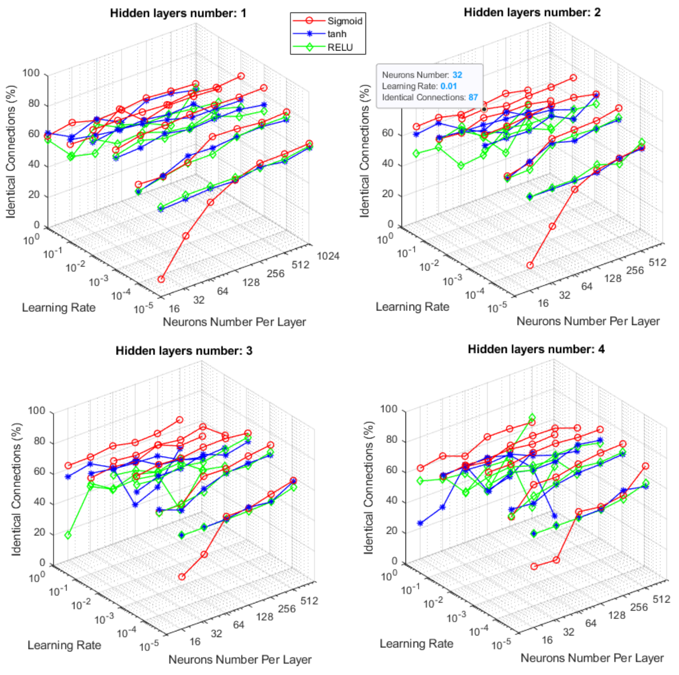


Fig. 14. Identical connections comparison with different hyperparameters used for training. The activation function for all hidden layers are RELU and only the activation function of the output is changed. As you can see, in general, sigmoid at the output layer gives better results. Two hidden layers with 32 neurons at each layer is the simplest architecture that gives a high identical connections of 87% as it is also shown in Fig. 6.

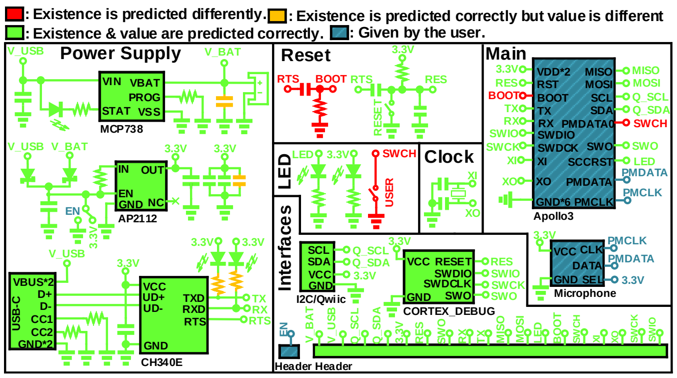

🟥: Existence is predicted differently.
🟧: Existence is predicted correctly but value is different
🟩: Existence & value are predicted correctly.
🟦: Given by the user.
Power Supply
Reset
Main
LED
Clock
Interfaces
MCP738
AP2112
CH340E
Apollo3
I2C/Qwiic
CORTEX_DEBUG
Microphone
Header
USB-C

Fig. 15. An example of a generated SBC schematic by FASCINET. Different colors show a comparison of the similarities between this FASCINET generated SBC and the human-designed one.

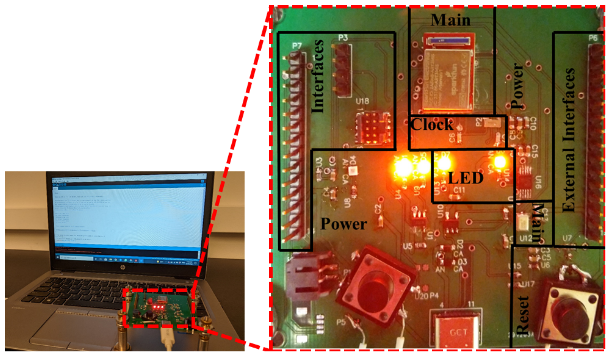


Fig. 16. The manufactured PCB of the SBC in Fig. 15. A simple blinking program is uploaded and run successfully on the SBC.

In addition to the DRC check, each individual connection in the schematics designed by FASCINET is compared with those in the human-designed ones, manually as well as by using a netlist comparison tool [49]. Table VII summarizes similarities between the generated peripheral circuits by FASCINET and the human-designed ones. Identical connections are defined as the portion of connections that are completely identical, meaning they are connected to the identical pins of the components which both the components and their values are correctly determined. For calculating the precision, recall and F1-Score, true-positive is defined as the number of connections that are connected to the same component in both tool and human designs without considering the component value. False-positive and false-negative are defined as the number of connections that exist in the tool or human designs, respectively but they do not exist in the other. It is noteworthy that the deviations of FASCINET designs from the human-designed references are mainly due to different design approaches, rather than functionality deficiencies.

Fig. 13 demonstrates the distribution of identical connections over the compared designs. The generated SBCs with a lower identical connections usually either contain new main components that have not been observed in the training set, or they have more than 4 main components, or they contain more than 70 components in total which makes them more complicated. It should be noted that in our test set with 400 SBCs, there are 40 designs that contain new main components that have not been observed in the training set. Between such designs, 25 of them contain main components that have the same category, e.g. microcontroller, with the main components of some designs in the training set. The average identical connections for these designs is 77.5% while it is 71.1% for the other 15 designs with totally new main components. Fig. 14 shows the identical connections while different hyperparameters are applied for training.

The output of FASCINET is an schematic in the industry standard format. Fig. 15 depicts an example of such a generated SBC schematic by FASCINET with 40 determined peripheral components and more than 120 connections. The blue hatched (main) components and connections are indicated by the user as the input. To obtain their information, both methods of describing specifications and directly inputting their datasheets have been tested. The final generated schematics are completely the same in these two approaches. However, it takes around 2 minutes on average to generate an SBC schematic when the specifications of the main components are inputted by the user, while it is around 5 minutes when a datasheet is inputted directly. It goes without saying that both methods are drastically faster than a human designer. The reason that the second method takes more time is because the datasheet scrubber is used rather than the COTS DB's search engine. In Fig. 15, the green color depicts components and connections that both the components and their values are determined correctly by FASCINET, meaning they are the same with the human-designed one. Orange is used to show determined components with differing values. Finally, red is used to show the components and connections that differ from the human design. Red and orange components do not mean that the tool has made an incorrect design. Rather, they just show a difference from how a human designer would make the circuit.

Fig. 16 shows a snapshot of the manufactured PCB of Fig. 15. A simple blinking program is uploaded and run successfully on the SBC. The SBC layout is generated by passing the Fig. 15 schematic file to an automatic PCB layout generator software [50], [51]. For this purpose, the automatic routing feature by specifying more width for power traces than normal signals using the directives of the layout generator software is used while the placement has been done manually.

VIII. Conclusion

This work presents a novel automatic SBC generator tool, FASCINET, which uses the HDNN-based approach for deciding the required peripheral components, determining their values and connections. As a part of this work, to build an efficient database of reusable components, a fully automated approach to extract the functionality and specifications of components from datasheets is proposed. The automatic category recognizer is trained and tested with more than 770,000 PDF documents within the COTS DB. It achieves an accuracy greater than 95%. The table extractor achieves a precision greater than 96% on a randomly selected testing set. The automated SBC generator tool is validated by comparing the netlist of the generated SBCs with more than 400 arbitrary human-designed ones. The required time for an SBC generation is reduced to $2-5$ minutes on average using FASCINET. The PCB layout of a sample SBC designed by FASCINET is also successfully generated.

Acknowledgment

This material is based on research sponsored by Air Force Research Laboratory (AFRL) and Defense Advanced Research Projects Agency (DARPA) under agreement number FA8650-18-2-7844. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The authors also would like to thank Morteza Tavakoli Taba and Mohammad-hasan Fayazi for their helpful discussions.

References

[1] N. Alee, M. Rahman, and R. Ahmad, "Performance comparison of single board computer: A case study of kernel on arm architecture," in *2011 6th International Conference on Computer Science & Education (ICCSE)*. IEEE, 2011, pp. 521–524.

[2] J. Doerr, "Low-cost microcomputing: The personal computer and single-board computer revolutions," *Proceedings of the IEEE*, vol. 66, no. 2, pp. 117–130, 1978.

[3] S. J. Kopec Jr, Y.-f. Chan, and R. F. Hartmann, "Programmable interface for computer system peripheral circuit card," May 5 1992, uS Patent 5,111,423.

[4] N. Patin, *Power Electronics Applied to Industrial Systems and Transports, Volume 1: Synthetic Methodology to Converters and Components Technology*. Elsevier, 2015.

[5] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, "Object recognition with gradient-based learning," in *Shape, contour and grouping in computer vision*. Springer, 1999, pp. 319–345.

[6] A. Ahmad, M. F. Roslan, and A. Amira, "Throughput, latency and cost comparisons of microcontroller-based implementations of wireless sensor network (wsn) in high jump sports," in *AIP Conference Proceedings*, vol. 1883, no. 1. AIP Publishing LLC, 2017, p. 020010.

[7] R. Dreslinski *et al.*, "Fully-autonomous soc synthesis using customizable cell-based synthesizable analog circuits," University of Michigan Ann Arbor United States, Tech. Rep., 2019.

[8] Adafruit Industries, https://www.adafruit.com/, last accessed 2020-05-08.

[9] SparkFun Electronics, https://www.sparkfun.com/, last accessed 2020-05-08.

[10] A. Tengli, Y. Yang, and N. L. Ma, "Learning table extraction from examples," in *Proceedings of the 20th international conference on Computational Linguistics*. Association for Computational Linguistics, 2004, p. 987.

[11] K. Nishida, K. Sadamitsu, R. Higashinaka, and Y. Matsuo, "Understanding the semantic structures of tables with a hybrid deep neural network architecture," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017.

[12] M. O. Perez-Arriaga, T. Estrada, and S. Abad-Mota, "Tao: system for table detection and extraction from pdf documents," in *The Twenty-Ninth International Flairs Conference*, 2016.

[13] Y. Liu, K. Bai, P. Mitra, and C. Giles, "Searching for tables in digital documents," in *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, vol. 2. IEEE, 2007, pp. 934–938.

[14] S. R. Qasim, H. Mahmood, and F. Shafait, "Rethinking table recognition using graph neural networks," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2019, pp. 142–147.

[15] S. Schreiber, S. Agne, I. Wolf, A. Dengel, and S. Ahmed, "Deepdesrt: Deep learning for detection and structure recognition of tables in document images," in *2017 14th IAPR international conference on document analysis and recognition (ICDAR)*, vol. 1. IEEE, 2017, pp. 1162–1167.

[16] S. S. Paliwal, D. Vishwanath, R. Rahul, M. Sharma, and L. Vig, "Tablenet: Deep learning model for end-to-end table detection and tabular data extraction from scanned document images," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2019, pp. 128–133.

[17] W. P. Birmingham, A. Brennan, A. P. Gupta, and D. P. Siewiorek, "Micon: A single-board computer synthesis tool," *IEEE Circuits and Devices Magazine*, vol. 4, no. 1, pp. 37–46, 1988.

[18] M. A. Curran, R. J. Diehl, G. A. Peck, and K. L. Rahaman, "Single board computer quotation and design system and method," May 24 2005, uS Patent 6,898,580.

[19] W. P. Birmingham and D. P. Siewiorek, "Micon: a knowledge based single board computer designer," in *21st Design Automation Conference Proceedings*. IEEE, 1984, pp. 565–571.

[20] F. Anderson, T. Grossman, and G. Fitzmaurice, "Trigger-action-circuits: Leveraging generative design to enable novices to design and build circuitry," in *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, 2017, pp. 331–342.

[21] J.-Y. Lo, D.-Y. Huang, T.-S. Kuo, C.-K. Sun, J. Gong, T. Seyed, X.-D. Yang, and B.-Y. Chen, "Autofritz: Autocomplete for prototyping virtual breadboard circuits," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 2019, pp. 1–13.

[22] Circuito, https://www.circuito.io/, last accessed 2021-10-15.

[23] A. M. Mehta, J. DelPreto, B. Shaya, and D. Rus, "Cogeneration of mechanical, electrical, and software designs for printable robots from structural specifications," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 2892–2897.

[24] F. Slomka, M. Dorfel, R. Munzenberger, and R. Hofmann, "Hardware/software codesign and rapid prototyping of embedded systems," *IEEE Design & Test of Computers*, vol. 17, no. 2, pp. 28–38, 2000.

[25] J. Bachrach, D. Biancolin, A. Buchan, D. W. Haldane, and R. Lin, "Jitpcb," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 2230–2236.

[26] R. Lin, R. Ramesh, C. Chi, N. Jain, P. Dutta, and B. Hartmann, "Supporting circuit design with a block-based, generator language," in *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020, pp. 1–8.

[27] EDASolver, "Edasolver - automatic component selection and pin matching," http://edasolver.com/, last accessed 2021-10-15.

[28] M. Fayazi, Z. Colter, E. Afshari, and R. Dreslinski, "Applications of artificial intelligence on the modeling and optimization for analog and mixed-signal circuits: A review," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2021.

[29] J. G. Shanahan and N. Roma, “Improving svm text classification performance through threshold adjustment,” in *European Conference on Machine Learning*. Springer, 2003, pp. 361–372.

[30] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation,” in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[31] S. Hochreiter, “The vanishing gradient problem during learning recurrent neural nets and problem solutions,” *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998.

[32] Z. Colter *et al.*, “Tablext: A combined neural network and heuristic based table extractor,” *arXiv preprint arXiv2104.11287*, 2021.

[33] R. Smith *et al.*, “Tesseract ocr: Open source scientific tools for python,” https://github.com/UB-Mannheim/tesseract, last accessed 2020-05-15.

[34] Digi-Key Electronics, https://www.digikey.com, last accessed 2020-05-15.

[35] S. Chickerur, A. Goudar, and A. Kinnerkar, “Comparison of relational database with document-oriented database (mongodb) for big data applications,” in *2015 8th International Conference on Advanced Software Engineering & Its Applications (ASEA)*. IEEE, 2015, pp. 41–47.

[36] B. Dipina Damodaran, S. Salim, and S. M. Vargese, “Mongodb vs mysql: a comparative study of performance in super market management system,” *International Journal of Computational Science and Information Technology (IJCSITY)*, vol. 4, no. 2, pp. 31–38, 2016.

[37] G. Alfian, M. Syafrudin, and J. Rhee, “Real-time monitoring system using smartphone-based sensors and nosql database for perishable supply chain,” *Sustainability*, vol. 9, no. 11, p. 2073, 2017.

[38] M. B. Arethiya and M. Garg, “Comparative analysis of simple and aggregate query in mongodb,” *International Journal of Advance Research, Ideas And Innovations In Technology*, vol. 4, no. 3, 2018.

[39] Y. Lou, F. Ye *et al.*, “Research on data query optimization based on sparksql and mongodb,” in *2018 17th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)*. IEEE, 2018, pp. 144–147.

[40] A. Gulli and S. Pal, *Deep learning with Keras*. Packt Publishing Ltd, 2017.

[41] Texas Instruments, “Lm117,” https://www.ti.com/lit/ds/symlink/lm117.pdf, last accessed 2020-05-10.

[42] Diodes Incorporated, “Ap2112,” https://www.diodes.com/assets/Datasheets/AP2112.pdf, last accessed 2020-05-10.

[43] R. J. Smith and R. C. Dorf, *Circuits, devices and systems: a first course in electrical engineering*. John Wiley & Sons, 1991.

[44] M. B. Alawieh, F. Wang, and X. Li, “Efficient hierarchical performance modeling for analog and mixed-signal circuits via bayesian co-learning,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 12, pp. 2986–2998, 2018.

[45] C. Sun, X. Qiu, Y. Xu, and X. Huang, “How to fine-tune bert for text classification?” in *China National Conference on Chinese Computational Linguistics*. Springer, 2019, pp. 194–206.

[46] M. Aristarán *et al.*, “Tabula: Open source scientific tools for python,” https://github.com/tabulapdf/tabula, last accessed 2020-05-15.

[47] B. K. Vasan, D. J. Chen, and R. L. Geiger, “Adc integral non-linearity testing with low linearity monotonic signals,” in *2011 IEEE International Instrumentation and Measurement Technology Conference*. IEEE, 2011, pp. 1–5.

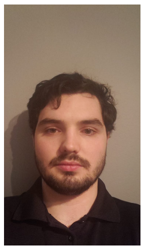
[48] N. S. Kim *et al.*, “Leakage current: Moore’s law meets static power,” *computer*, vol. 36, no. 12, pp. 68–75, 2003.

[49] Cadence, “Cadence design systems,” https://www.cadence.com/en_US/home.html, last accessed 2021-07-20.

[50] Altium, https://www.altium.com/, last accessed 2020-05-27.

[51] Eagle, https://www.autodesk.com/products/eagle/overview, last accessed 2020-05-27.

Morteza Fayazi was born in Tehran, Iran, in 1994. He received the B.Sc. major degree in electrical engineering and minor degree in computer science from Sharif University of Technology (SUT), Tehran, Iran, in 2017. He Also received the M.S. degree in electrical engineering and computer science in 2020 from University of Michigan, Ann Arbor, MI. He is currently working toward the Ph.D. degree at University of Michigan, Ann Arbor, MI. His research interests include circuit design automation, machine learning, software development and computer architecture.



Zach Colter was born in Michigan, United States in 1997. He recieved both the B.Sc. degree, in computer engineering, and M.S. degree, in electrical computer engineering, from the University of Michigan. The degrees were acquired in 2019 and 2020 respectively.

Zineb Benameur El Youbi received the B.Sc. and M.S degree in Computer Networks and Multimedia Communication from the University of Grenoble, France, in 2009. She Also received the M.S. degree in Computer Science and Engineering in 2020 from University of Michigan, Ann Arbor, MI. Her research interests include reliable software development and Machine Learning.

Javad Bagherzadeh received his B.Sc. and M.Sc. degrees in electrical engineering from Shahid Beheshti Univeristy and Sharif University of Technology in Tehran, Iran, 2010 and 2012 and a PhD degree from The University of Michigan in 2021. He is currently a post doctoral researcher at the University of Michigan, Ann Arbor, MI. His research interests interests lie primarily in architectures that enable emerging device technologies and applications.

Tutu Ajayi received a B.Sc. degree in electrical engineering from Texas A&M University in 2008 and a PhD degree from The University of Michigan in 2021. He is currently a post doctoral researcher at the University of Michigan, Ann Arbor, MI. His research interests interests lie primarily in cross-layer computer architecture, with an emphasis on realizing research into ASIC or FPGA platforms.

Ronald Dreslinski received the BSE degrees both in electrical engineering and computer engineering, and the MSE and PhD degrees in computer science and engineering from the University of Michigan, Ann Arbor, Michigan. He is currently an assistant professor of computer science and engineering with the University of Michigan. His research focuses on architectures that enable emerging low-power circuit techniques.