Design Space Exploration for Board-level Circuits: Exploring Alternatives in Component-based Design

Richard Lin
richardlin@ucla.edu
University of California, Los Angeles
Los Angeles, USA

Rohit Ramesh
rkr@berkeley.edu
University of California, Berkeley
Berkeley, USA

Parth Pandhare
parthpandhare@ucla.edu
University of California, Los Angeles
Los Angeles, USA

Kai Jun Tay
kaijuntay687@ucla.edu
University of California, Los Angeles
Los Angeles, USA

Prabal Dutta
prabal@berkeley.edu
University of California, Berkeley
Berkeley, USA

Björn Hartmann
bjoern@berkeley.edu
University of California, Berkeley
Berkeley, USA

Ankur Mehta
mehtank@ucla.edu
University of California, Los Angeles
Los Angeles, USA

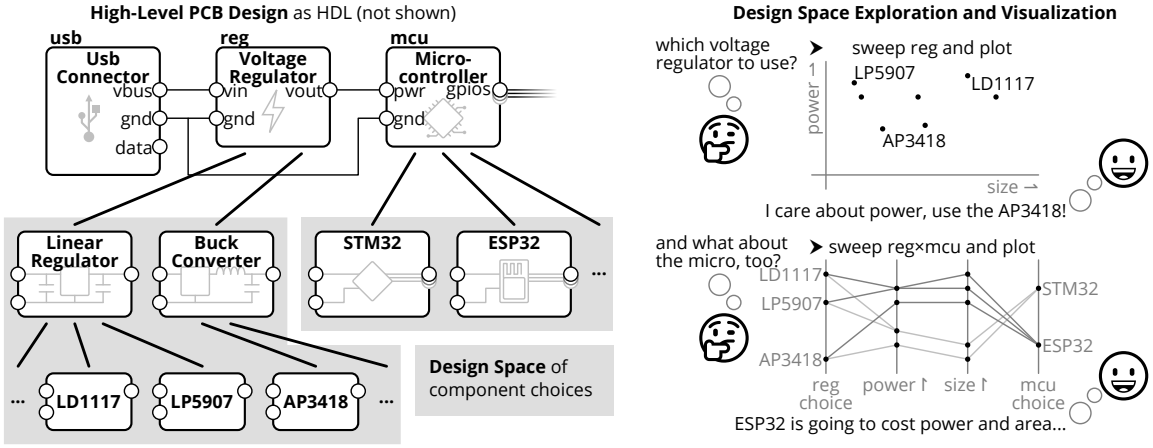

Figure 1: Overview of our approach to design space exploration for electronics design. The user designs their board at the system architecture level of abstraction like represented in the top left, using a hardware description language (HDL) [13] supporting library blocks. Some of these blocks are ambiguous (e.g., microcontroller and voltage regulator) with alternatives defined in the library. The user can then select these degrees of freedom to explore, and additionally plot objective functions like current draw and component board area to aid in making a choice using both 2-axis scatterplots (top right) and n-axis parallel coordinate plots (bottom right).

ABSTRACT

While recent work explores novel tools to make electronics and device design easier and more accessible, these tend to be either highly automated (great for novices, but limiting for more advanced users) or highly manual (suitable for experts, but imposes a higher skill barrier to entry). In this work, we examine a middle ground: user-guided design space exploration to bridge an intuitive-but-ambiguous high-level representation to a fully-specified, fabrication-ready circuit. Our system helps users understand and make design choices by sweeping the design space of alternatives for electronics parts (e.g., choice of microcontroller), marking invalid options, and plotting points to visualize trade-offs (e.g., for power and size). We discuss the overall system and its structure, report on the results of a small but in-depth user study with participants from a wide range of electronics backgrounds, and draw insights on future directions for improving electronics design for everyone.

CHI '24, May 11–16, 2024, Honolulu, HI, USA

ACM ISBN 979-8-4007-0330-0/24/05
https://doi.org/10.1145/3613904.3642009

CCS CONCEPTS

• **Hardware → PCB design and layout**; *Software tools for EDA.*

KEYWORDS

electronics design, printed circuit board (PCB) design, design space exploration

ACM Reference Format:
Richard Lin, Rohit Ramesh, Parth Pandhare, Kai Jun Tay, Prabal Dutta, Björn Hartmann, and Ankur Mehta. 2024. Design Space Exploration for Board-level Circuits: Exploring Alternatives in Component-based Design. In *Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI '24), May 11–16, 2024, Honolulu, HI, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3613904.3642009

1 INTRODUCTION

Electronics are integral to many modern devices, but designing the custom electronics to enable these custom devices is a non-trivial process. Although prototyping is a well-supported process given the availability of (relatively) easy to use development boards and much recent work on tools for novices, the capability ceiling is low and non-trivial devices must be built on custom printed circuit boards (PCBs) [11].

While recent work has examined better ways to support PCB design, most fall under either highly automated approaches (e.g., circuit synthesis) geared towards novices, or library-based but still manual approaches (e.g., block-based modular design) offering more control. Yet, the strength of each approach is also the drawback of the other: highly automated tools may not offer enough control for experts who may have specialized requirements, while more manual tools rely on user expertise to resolve design choices - even if the very-low-level electrical engineering details are encapsulated by libraries. An example for both cases is the choice of microcontroller: an expert may have specialized requirements outside the understanding of the tool and need to select a specific part, while a novice may not have the context to make a valid choice on their own. Ideally, we could have PCB tool for a wide range of use cases, towards the "low floors, high ceilings" ideal for creativity support tools [32] with the resulting benefits of a common platform like a wide base of community libraries.

One possible hybrid solution is *design space exploration* (DSE), where instead of the tool or user being solely responsible for decisions, the tool helps the user understand the design space and choices. A rare instance of this approach in board-level electronics is Trigger-Action-Circuits [3], a breadboard circuit synthesis tool for novices. Its alternatives explorer allows the user to choose between multiple satisfying designs, ultimately enabling a finer degree of control. While this feature was very basic and not the primary focus, its user study suggested that future work could further help users compare choices.

In this work, we focus in on DSE for PCBs and built a prototype tool leveraging our prior work on a type system of electronics parts that implicitly defines a design space [13]. The user starts by creating a high-level design using abstract parts like "microcontroller" and "voltage regulator". Each possible choice (e.g., part number for a microcontroller) is a design point, and the system sweeps over the design space, testing each point. The resulting points can be plotted visually, both indicating validity (e.g., can this microcontroller work in this design) and objective functions (e.g., the price or area if this microcontroller is used), and ultimately helping the user make an informed choice.

Our hope is that this can be a good solution for a broad audience of electronics designers: easily interpreted context to help novices make a valid choice, but the flexibility for experts who have design considerations beyond the system's model. As such, we take a middle-out approach for the target audience: focusing on those with some level of electronics expertise (loosely, makers), but still being degrees of useful for novices and experts. This prototype tool then allows us to conduct a user study to understand how this concept works in practice and build ideas for future work.

Overall, we seek to answer:

- How DSE can be useful for electronics design, especially as a technique for a wide audience from levels of novice to degrees of expert.
- How users of different skill levels can make use of this capability, and what are the strengths and shortcomings.

And we contribute:

- A prototype **DSE tool for PCB design**[1], using visualization and computed objective functions to help users understand the design space and resolve design details.
- A preliminary **user study** of how a range of PCB designers (from novices to professionals) use this tool, in a fairly realistic setup where participants bought their own projects. We found that participants were able to use DSE to find viable (and even optimized) designs among alternatives, while interview feedback suggests directions for future work on electronics design tools in general and improving usability specifically.

In the rest of this paper, we go over related work, introduce our system, describe our user study and results, and conclude with a discussion on insights from the study and ideas for future work.

2 RELATED WORK

We build upon prior work in electronics design and both foundational work on design space exploration and its application to other domains.

2.1 Electronics Design

In electronics in general, there is a large amount of work on supporting novices, especially around breadboard prototyping [6, 12, 41]. While our work focuses on PCB design, some of this prototyping work has aspects of mixed-initiative interactions and design space exploration. AutoFritz [18] extends the Fritzing [12] mixed schematic / breadboard view with an autocomplete function, suggesting circuit additions but ultimately keeping the user in control. Trigger-Action-Circuits [3] produces a breadboarding diagram, but synthesizes a circuit from a behavioral specification (e.g., button-press, measure brightness, light-up). Beyond returning just one result, it presents users with the design space of multiple generated circuits in a spreadsheet view – though its user study indicates that users were unsure how to use these results. Our work expands on this idea with a dedicated DSE tool that helps the user understand

[1]Open source at https://github.com/BerkeleyHCI/edg-ide, archival version also included in supplementary materials.

the design space to make choices, and in the more intermediate context of PCB design.

For PCB design specifically, there have been a few observational studies of current practices. One study [21] observed novices over a six-week workshop in which they learned to prototype and ultimately design a custom PCB. While most participants got to functional hardware, they also noted challenges with selecting suitable components and subcircuit blocks. The other [16] was an interview study with a wide range of PCB designers on current design practices. One finding was that design choices were made throughout the design process, commonly outside PCB design tools. Our work here attempts to integrate many of those choices into a unified tool with a sweepable design space.

2.1.1 PCB Design Tools. In the larger picture, PCB design tools fall into a larger space of creativity support tools [31–33]. One useful framing, which we will reference in the rest of this paper, is the idea of *low floor* (easy to get started for novices), *high ceiling* (useful for experts), and *wide walls* (wide range of functionality) [32].

There has been some work on automated synthesis of circuit designs for PCBs. Both Embedded Device Generation [27] and Echidna [22, 23] have users specify a partial design with parts like sensors and motors. The system then completes the rest of the design through interface-driven synthesis, such as by adding infrastructural components like voltage translators and microcontrollers. Appliancizer [8] does a similar operation, but with the input as a web page, where HTML elements like buttons are mapped to hardware like physical switches. Outside academic work, circuito.io [4] and EDASolver [7] similarly complete a design from a list of peripherals specified by the user. All of these have low floors, but also a low ceiling and narrow walls given the lack of fine-grained user control and extensibility.

On the other end of the spectrum for recent PCB design work are subcircuit library-based tools. Unlike synthesis tools, these leave the part choices up to the user, but they can still make design easier and faster through the use of higher-level blocks (e.g., ATMega328P microcontroller subcircuit, instead of a discrete chip and supporting passives) from a library of parts. Sparkfun À La Carte [35] and Upverter Modular Design [39] (previously Gumstix Geppetto) are two commercial examples. However, these have a higher floor than circuit synthesis as the user must understand the high-level system architecture and parts selection.

The other thread of work in this category has been our open-source hardware description language (HDL) for PCB design [13, 14] and associated mixed textual / graphical block diagram IDE [17]. In particular, the HDL approach allows users to build custom libraries with subcircuit *generators* – code defining how to implement the lower-level subcircuit details from higher-level user parameters (e.g., size a resistive divider given a ratio). A type system over parts also supports *abstract*, or generic, components (e.g., the abstract microcontroller specifies an interface that any microcontroller can be substituted in to) and implicitly defines a design space, though users must still manually specify choices. The support for user-defined libraries provides a higher ceiling and wider walls, though like the commercial library-based tools the user still must understand the system architecture and parts selection.

We build on top of this work with a tool to sweep through, visualize, and meaningfully compare alternatives across this implicitly defined design space. We hope that this can lower the floor (without lowering the ceiling) by helping novices through parts selection, while also widening the walls for everyone as generally useful functionality.

Table 1 summarizes prior work in electronics design tools.

2.1.2 Chip Design. Although chip design is a form of electronics design, it is significantly different from board design in terms of design practices. Overall, chip design tends to be a tradeoff between speed, energy, and cost using highly automated methods like genetic algorithms [26] – though some work does examine human-in-the-loop design space visualization [37]. As chip simulation models are computationally expensive, much work focuses on optimizations such as with simplified models [10, 28]. In contrast, our work is based on rougher static models (e.g., maximum current draw) found in component datasheets, which are computationally inexpensive to work with.

2.2 Design Space Exploration

We also draw upon a significant body of work on design space exploration – though our focus is on its application to electronics.

2.2.1 Interfaces. Prior work has explored visualization approaches to supporting users in design space exploration. A thread of work examines design space exploration as design-by-shopping, where choices are framed as comparison shopping, including applications to car selection [2] and spacecraft design [1]. As both found parallel coordinates plots (a plot type supporting simultaneous visualization of arbitrary number of axes) to be useful, we implemented support for it.

For high-dimensional spaces (exploring many variables), dimensionality reduction techniques [40] can help produce a usable visualization by transforming the data. Philosophically, we choose to focus on more direct and straightforward visualizations, though future work can examine synthetic visualizations.

The *single-state document model* [38], where design tools require the design to be in one state at any time, is not suited for nonlinear creative processes with exploration of alternatives. Proposed alternatives have included improved preview support for alternatives [38], parallel creation and control over alternatives [19], and git-inspired branching and merging of versions [44]. While our underlying HDL is constrained to the single-state document model, abstract superclasses allow the single document to simultaneously represent all alternatives, while the design space exploration tool allows users to down-select to a concrete design point.

2.2.2 Application Domains. While design space exploration has been applied to many domains, the characteristics of current board design practices make direct applications of prior work challenging – like for chip design discussed above.

Much work exists on design space exploration for mechanical design, such as on visualization to compare design points [20] and even direct manipulation in virtual reality (VR) [9]. On the algorithms side, a thread of work examines interpolating between points [30] and calculating an approximate *Pareto frontier* [29] (roughly, the best designs possible without making trade-offs).

Work	Design abstraction	Automation	Availability	User Libraries
Trigger-Action-Circuits [3]	Behavioral	Synthesis		Yes [a]
Echidna [22]	Partial design	Synthesis		
Appliancizer [8]	Peripherals list [b]	Synthesis		
Embedded Device Generation [27]	Peripherals list	Synthesis	Open-source	
circuito.io [4]	Peripherals list	Synthesis	Web app	
EDASolver [7]	Peripherals list	Synthesis	Web app	Yes [c]
AutoFritz [18]	Breadboard	Autocomplete		
Sparkfun À La Carte [35]	System architecture	High-level design	Web app [d]	
Upverter Modular Design [39]	System architecture	High-level design	Web app	Yes
Polymorphic Blocks [13]	System architecture [e]	High-level design	Open-source	Yes [f]
This work	System architecture	DSE	Open-source	Yes [f]

[a] Components defined in XML
[b] As a webpage, with elements mapped to hardware
[c] Components library is source-available on GitHub
[d] Requires purchase of fabrication service to get board design files
[e] As HDL, supporting generators for user-defined design automation in library construction
[f] All library blocks defined in user-facing HDL and open-source

Table 1: Summary of recent work on electronics design tools. In terms of the creativity support tools literature [32], we classify the behavioral abstraction as very-low floor, partial design and peripherals list as low floor, breadboard with autocomplete as low-medium floor, system architecture as medium floor, and system architecture with DSE as low-medium floor. We classify synthesis tools as having a low ceiling because of the lack of fine-grained user control, all the other tools can be high ceiling with either sufficient libraries or support for user-defined libraries.

While impressive, these algorithms mostly rely on a continuous parameter space (e.g., length of a part, which is a real number), compared to the mostly discrete design spaces (e.g., between part numbers) for PCBs. Additionally, we note that mechanical design has a few well-known and straightforward-to-calculate objective functions (e.g., minimize mass), whereas we seek to understand the important objective functions for electronics.

For cyber-physical systems, often a complex composition of electrical, mechanical, and controls domains, OpenMETA [36] attempts to integrate these domains as a meta-modeling platform. Like our work, it has a block diagram model and supports a design space sweep with results visualization [25, 36], though unlike our work the design space is user-defined on a per-block basis instead of embedded in the library as part of a component type hierarchy. We additionally explore how this idea work in practice and for electronics design through a user study.

3 EXAMPLE SYSTEM WALKTHROUGH

In this section, we introduce our system by walking through how a user might use design space exploration to design an example device: a USB-powered, microcontroller-controlled LED circuit. This provides a coherent overview of the system and its intended usage as a whole, while a more complete description of the system's capabilities and architecture follows in the next section.

3.1 High-level Design

As design space exploration requires a design space to search over, we build our system on top of prior work on an open-source HDL for electronics design [13] that supports high-level design with library blocks like "voltage regulator" and "microcontroller". These blocks are abstract (themselves having no implementation), and the concrete choices of each block (e.g., the microcontroller could either be a STM32 chip or ESP32 module) forms the design space.

The user starts by writing this high-level design HDL, optionally using schematic-like graphical actions which generate code [17]. Although HDL is required by the underlying design system and is part of the user interface, the DSE system operates mostly independently from it and the rest of this paper focuses on the conceptual block diagram model.

This example design consists of a USB connector, voltage regulator to drop the USB voltage, and microcontroller, as illustrated in Figure 1 top-left. Connected to the microcontroller would be a LED (not shown, connected to the microcontroller GPIO).

Compiling this design produces a block diagram like in Figure 2 top-left. As both microcontroller and voltage regulator are abstract parts, ideal models (e.g., a voltage regulator that hits its voltage specification exactly, or a microcontroller with infinite pins) are automatically substituted in with errors generated to require a choice from the user. A parameter browser panel (as in Figure 2 bottom-left) also enables the user to explore calculated values like current draw.

3.2 Searching Voltage Regulators

In this first scenario, where the user has already chosen a microcontroller, only the choice of the voltage regulator remains.

The user can start by defining the design space by right-clicking on the voltage regulator on the block diagram visualizer, and selecting "search subclasses" (as in Figure 2 top-left). The user can similarly define objective functions of interest, such as area and current draw, with the latter through the parameter browser (as in Figure 2 bottom-left). A design space configuration view (as in Figure 2 bottom-center) lists the selected search space and objective

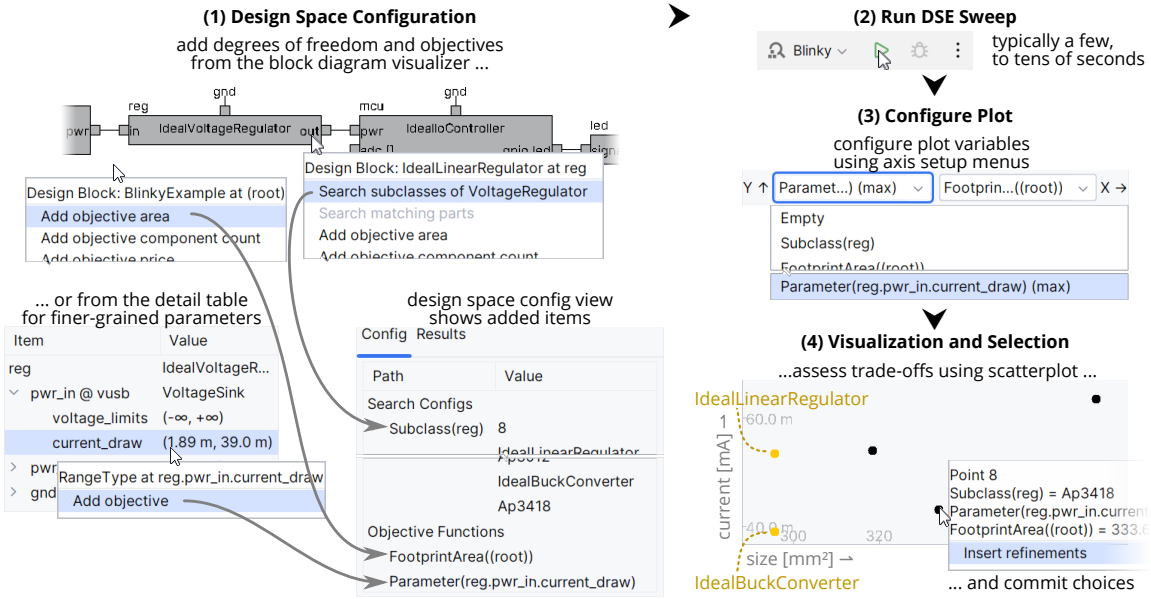

Figure 2: Example workflow for a design space search across voltage regulators. The user starts by (1) adding objectives like component area and current draw, and defining the design space like concrete subtypes for the voltage regulator, then (2) running the sweep. With results from the sweep, the user can then (3) configure the scatterplot visualization by setting the axes with objective functions like current draw and area. Each design is plotted on the scatterplot (4), color coded by validity. Black points represent valid designs, like the selected AP3418 converter. Brown points represent ideal components which provide a visual representation of a best-achievable case for a class of choices, like the ideal voltage regulator having zero area and perfect efficiency current draw. Red points (not shown) represent invalid designs, like a high-precision but low-current voltage regulator that cannot supply the needed power.

functions. Here, the space includes all subclasses of voltage regulators, including linear regulators (low-efficiency voltage step-down), buck converters (high-efficiency but complex voltage step-down), and boost converters (voltage step-up – not useful here).

With the search space defined, the user can run the design space sweep, which compiles all points in the design space and completes in about 10 seconds. These compiled design points are presented as both a table (not shown) and a scatterplot (as in Figure 2 bottom-right) which the user can configure the axes for (as in Figure 2 center-right). Plotting current draw vs. area, the plot shows the trade-off between size and efficiency: while the buck converter (bottom point) has low current draw, it takes up larger board area compared to some linear regulators (top points).

Once the user makes a choice, they can right-click the point (as in Figure 2 bottom-right) to insert the choice as a *refinement* in their HDL. With a complete design specified, the user can also generate the netlist file of components and connections to import into a mainstream PCB layout tool for physical design and ultimately fabrication.

3.3 Searching Voltage Regulators and Microcontrollers

In this second scenario, the user also adds the microcontroller to the search space, to search across all possible combinations (Cartesian product) of microcontrollers and voltage regulators. This new sweep covers some 200 points, and finishes in about a minute. Similar to the voltage regulator-only case, results are listed in the table view and can also be displayed on the scatter plot.

For these higher dimensional spaces, we also provide a parallel coordinates plot, detailed in Figure 3. While scatter plots are limited to two axes, parallel coordinate plots can support an arbitrary number of axes, displayed side-by-side and parallel to each other. Each design point has (at most) one point on each axis corresponding to its value for that axis, and a polyline through all those points allows visually correlating adjacent axes. Here, the user can simultaneously plot the microcontroller choice, current draw, component area, and voltage regulator choice.

Larger design spaces can produce a complex plot with many overlapping points and lines as in Figure 3 top, though insights may still be readily available. More practically, the user can select points to dim out the rest, for example clicking the ESP32 on the microcontroller axis to arrive at Figure 3 middle. Navigation features like zooming allows nearby points to be disambiguated, for example spreading out the current draw to get to Figure 3 bottom. Similar to the scatterplot case, once a single design point is selected, the refinement HDL can be inserted, here both the choice of microcontroller and voltage regulator.

4 SYSTEM DESCRIPTION

Our system consists of the user-defined design space and objective functions, the results table and plots, and the design space sweep process. Overall, the guiding philosophy is to support user-guided

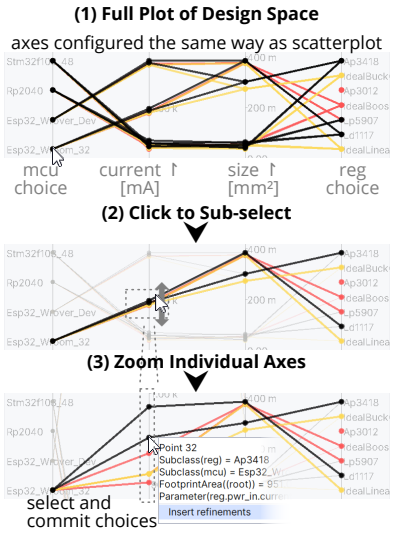

Figure 3: Example workflow for the parallel coordinates plot, for a search across all combinations of voltage regulators and microcontrollers. After configuring the search space and running as in Figure 2 (1) and (2), the designs can be visualized on a parallel coordinates plot. After selecting the axes (not shown), the plot shows the entire design space in (1). Even here, a few trends can be inferred: the boost converters (middle on the rightmost axis) are all invalid, and some microcontrollers (leftmost axis) draw much more current. Clicking a point, such as the ESP32 on the microcontroller axis in (1) selects only designs going through that point, dimming out everything else as in (2). Individual axes can be zoomed in (and out) to disambiguate close points, such as expanding the current draw axis as in (3) to select the lowest current draw design.

DSE, emphasizing predictability and control over more magical automation. This is implemented on top of the open-source IDE plugin [17] for the HDL.

4.1 Design Space Definition

The overall design space for any sweep operation is a user-specified list of choices for some variables. From this, the system generates the full design space using the Cartesian product of variable choices. Users can also choose to sequentially search over different parts, avoiding the exponential computational cost of a combinational search if combinational interactions are not important.

These variables are available for design space definition:

- **Subclass**: as shown in the example, the user can sweep all subclasses of an abstract block.
- **Parameters**: the user can specify choices for using free text entry. Parameters are variables in the underlying HDL and are often used to define the specification for some block, e.g., the resistance of a resistor. Parameter choices can apply on a per-instance or per-block-class basis, the latter allowing, for instance, to test the effects of minimum package size for SmdStandardPackage[2] devices on a board-wide level.

[2]0402, 0603, etc. series

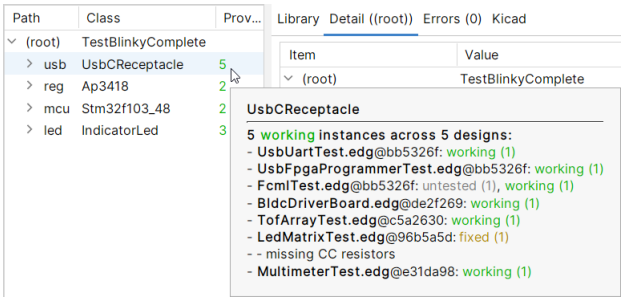

Figure 4: Example tested status for the USB-C receptacle. The green 5 under the Proven column indicates that there have been 5 tested-working instances of this block since the last major change. The tooltip provides a more detailed listing of the designs this block has been used in, including the 5 tested-working instances, the broken one prior that has since been fixed (orange), and one untested instance (grey).

- **Parts**: while complex parts are represented in the type hierarchy (e.g., the Ap3418 buck converter in the example), more generic parts are automatically selected by matching parameters (e.g., resistor based on resistance) from a parts table. Often, many parts satisfy the user's specification and one is chosen arbitrarily, but this allows sweeping the entire space of compatible parts.

4.2 Objective Functions

The user can also specify values of interest to be displayed on the results table or plotted graphically:

- **Parameters**: as general variables in the underlying HDL, parameters include calculated outputs like a voltage regulator's current draw in the context of the system[3].
- **Component area**: the total area[4] of all components in a block, as a rough gauge of board size.
- **Component count**: the number of components in a block as a rough gauge of complexity.
- **Price**: the total cost[5] of components in a design.
- **Unproven**: the number of blocks that haven't been tested in hardware before as a rough gauge of design risk. See Section 4.3 for details.

4.3 Proven Data

As prior work [13, 16] found that users had concerns about the correctness of libraries they didn't write, we wanted to capture some notion of this risk in our objective functions by identifying which blocks have been built and tested[6]. We categorized blocks into working, fixed, broken, and untested, where both broken and untested contribute to the unproven count objective function.

[3]There currently isn't a distinction between input and output parameters, so it is up to the user to understand the meaning of each parameter – though this could be improved in future work or through documentation.
[4]Specifically, the footprint's courtyard area.
[5]Based on the April 2022 JLCPCB parts table with some supplemental rows.
[6]Based on a database of prior example designs we built.

Additionally, the proven statuses for each prior design are viewable as a popup on the block itself, as a more detailed breakdown of design risk.

4.4 Results and Plots

The results table and plots are mostly as described in the example: the results table shows a list of all the design points with a column for each objective, while the scatter plot and parallel coordinate plot allow the user to select their axes from both the design space and the objective functions. Plots are initially scaled to the range of the data and zero, but the user can pan and zoom.

Designs with ideal components are shown in brown and can be used to show a theoretical maximum objective, e.g., the best-case power for the voltage regulators in Figure 2. Designs with errors (where the electronics model failed to validate, e.g., exceeding voltage limits) are shown in red and may be a prompt for the user to explore further, perhaps seeing if some constraints can be relaxed to achieve better outcomes. Some errors may prevent the design from compiling (e.g., using a boost converter to step down the voltage, which exceptions out) and are not shown as there is no data.

While numerical objectives are relatively straightforward for plotting, non-numeric types can also be plotted, rendered as evenly-spaced strings. Subclasses are plotted as strings, with related classes grouped together[7]. Range-valued data (intervals, used extensively throughout the HDL libraries to model tolerances and allowable ranges) requires the user to select the minimum, center, maximum, or tolerance.

4.5 Design Space Sweep

The design space sweep process is conceptually straightforward, testing all choices one-by-one and live-updating the results table and plot after each design point.

We optimize the sweep by saving and re-using a partially compiled design, instead of compiling every design from scratch. Since the design space defines which variables are being swept over, the design is first compiled with all of those variables blocked. For parameters, this means that the parameters and dependencies on it remain uncompiled[8]. For subclasses, this means compilation does not progress into that block, leaving it, its parameters, and any internal blocks uncompiled. Then, for each design point, the compiler state is copied, and the variables are unblocked and assigned a value from the design point. Only work that depends on the choice being tested needs to be performed on a per-design-point basis.

In practice, the speedup of this technique is heavily dependent on the design. For example, for the LED design and sweeping over both the voltage regulator and microcontroller (198 designs), with pre-compilation it takes 74.7s[9], and without pre-compilation it takes 125.9s. On the other hand, for a more complex multimeter example described in prior work [14] and sweeping over the microcontroller (11 designs), with pre-compilation it takes 37.0s, and without pre-compilation it takes 88.5s.

[7]Specifically, ordered by the depth-first traversal of the class hierarchy.
[8]While the original HDL paper [13] mentioned a constraint system, work afterwards has simplified the parameter system to use only directed assignments, e.g. $a <= b + c$.
[9]All results on a Ryzen 1700X, 48GB RAM, average of three runs.

5 USER STUDY METHODOLOGY

Because there is limited prior work in DSE for board-level electronics design, we ran a user study both to gauge the usefulness of this kind of user-in-the-loop DSE for electronics design and as a kind of design probe to better understand user needs to drive future work. Towards understanding needs, our study focuses on realism (ecological validity) with participants working on a project of their choice, though this is balanced with completing the project within a reasonable amount of time and the libraries available. As more initial work, we focused on gathering rich, in-depth data that could serve as a springboard for future work, though at the cost of a smaller participant pool.

5.1 Participants

We recruited six (3F) participants through a mix of personal referrals and recruitment messages with an electronics-focused student group. Overall, we wanted to sample for range across prior PCB design experience levels, with a center-mass around low-intermediates (beginning makers) who we believe these tools could be most useful for. Experts and advanced users round out the data.

Participants were required to have some familiarity with Python and PCBs (to avoid needing to spend excessive time on fundamental concepts), but did not actually need to have designed a board prior. Participants were compensated with gift cards at $30 an hour and the option to get their board made if they also completed the board layout (on their own time).

5.2 Structure

The study was conducted with as a mix of videoconference and in-person meetings, at the participant's choice. In both cases, participants accessed a fresh virtual machine (with the software pre-installed) through the remote desktop software X2Go. In the remote case, we asked participants to share their remote desktop window so we could watch their progress.

We started by working with participants to determine a project of reasonable scope given available libraries and containing some element suitable for design space exploration. The study itself begins with a written tutorial, introducing the HDL and walking through the design space exploration example in Section 3. After the tutorial, participants continued on to their project, optionally starting from the tutorial HDL as a base. While we did not record these sessions, the software logged some actions (including plot configuration changes and design space sweeps) and we took notes.

As we were more interested in participants' feedback on the high-level concepts and learning about their needs, we would answer any questions and offer suggestions. This allowed us to run a study with a realistic project on a design tool with a non-trivial learning curve and lacking the polished interface, documentation, and community resources (e.g., Stack Overflow) of more mainstream tools.

Afterwards, we concluded with a semi-structured interview. Topics included general free-form feedback and specific questions on design space exploration including comparisons to their current flows, what they wanted to explore (whether supported in our system or not), performance issues, and speculative comparisons to mainstream and synthesis tools. The overall framing was getting constructive feedback and learning about participants' workflows,

which was both consistent with our goals and helps limit acquiescence bias. These were audio-recorded (with participants' consent) and transcribed. Analysis was done with an open coding pass and iterative grouping of codes into related topics [42]. We then focused on issues that were most interesting in terms of implications for design for DSE tools – both how our tools were used, and limitations that future work could explore in more depth.

5.3 Libraries and Design Space

The design spaces that participants could explore were limited to the libraries we provided. The abstract subclasses (where a choice is required) are mainly the microcontrollers and voltage regulators described in the example in Section 3. Microcontrollers include discrete ICs like the STM32F103, radiofrequency modules like those based on the nRF52 and the ESP32, and development boards like the ESP32 with an onboard camera. Voltage regulators include several options for linear regulators, and buck and boost switching converters. While not a huge design space, this does cover many commonly used parts.

Where a default is automatically selected from a parts table (such as for generic components like resistors, capacitors, and FETs), users could also choose to search through alternative compatible parts.

6 USER STUDY RESULTS

Overall, our six participants had experience levels ranging from novices to professionals and worked on a variety of projects, all summarized in Table 2. On average, participants completed the tutorial in 1h 18m and spent 2h 43m on their projects. Interviews averaged 1h 48m.

6.1 Individual Observations

All participants completed their designs to a point where the HDL and its electronics model checks were error-free and we believed that the design would be functional if made. Projects are described below, with simplified block diagrams for all in Figure 5.

6.1.1 P01: High Voltage Sensor. P01 designed a high voltage sensor, with a microcontroller, opamp buffer, and high voltage resistive divider (requiring multiple resistors in series to limit the voltage across each resistor, unlike the classic two-resistor divider). The implementation of the divider was the main focus of the search process: choose resistor values that produce some target voltage division ratio, while loosely optimizing for full scale output range (output voltage as close to 3.3 volts as possible) and balancing power consumption and signal integrity (which are inversely correlated). P01 specified resistor values[10] for the DSE tool to search, while the system created the full cartesian product and the resistive divider library calculated the objective values and validated design correctness (e.g., for output voltage and resistor voltage ratings).

Uniquely, P01 also wrote a quick script to generate the search space for resistance. In mainstream flows, P01 would sometimes write scripts to search the design space, sometimes using Monte Carlo probabilistic methods for intractably large spaces.

[10]Roughly the E6 series of preferred values

6.1.2 P02: LCR Meter. P02 designed a LCR (inductance-capacitance-resistance) meter, based off an existing open-source design[11] using a single-chip analog frontend[12]. The DSE tool was used to select infrastructural components like the microcontroller and voltage regulator and used to search through some of the analog frontend parts. While P02 was able to select the most precise (tightest tolerance) resistor using the DSE tool and visualization, the parts libraries lacked the more detailed modeling for other parts like diode leakage current and noise immunity so those parts were chosen with existing parts tables and datasheets. However, on the user interface side, P02 noted that the electrical parameters were buried in lists of other data and difficult to find, and suggested instead a list of only objective functions sorted by most commonly used.

6.1.3 P03: Greenhouse Sensor. P03 designed a greenhouse sensor, a wireless device that could measure temperature, humidity, and light spectrum. P03 started from the tutorial HDL, but cleared the refinements before adding the application-specific blocks like sensors. Uniquely, the AS7341 spectral sensor also needs a 1.8v supply, which generated an error and required another voltage regulator.

Workflow-wise, P03 added and connected all the application blocks, leaving the ideal blocks in place and ignoring the corresponding compiler errors. Only after the design was done did P03 use DSE to select the choice of voltage regulators and microcontroller. P03 compared this design flow to traditional schematics:

> I need an MCU on this board, but not having to kind of commit yourself to one and then put all the other components on and then realize, oh, that's not the one I want, and then having to like rearrange everything afterwards is a definite time saver in terms of prototyping

6.1.4 P04: Sleep Tracker. P04 designed a sleep tracker, roughly architected off a class project report[13] which has a heart rate sensor, breath sensor (a variable resistance 710kOhm elastic cord), and accelerometer. P04's overall process was to insert blocks from the library, compile to see the errors, incrementally fix them, and repeat until no errors remained.

Notably, the sensor for the elastic cord (a resistive divider, with the fixed resistor defaulting to 47kOhm) flagged an error because the high output resistance of the sensor could cause signal integrity issues. While P04 started by (perhaps unrealistically) perturbing the sensor resistance range, a bit of prompting from us to reflect on the circuit being built led P04 to use the DSE tool to try different values for the fixed resistor and to look at the automatically calculated output voltage range. From here, P04 scanned a combination of sensor resistance ranges and fixed resistor values (1kOhm to 50kOhm in about 10 steps) and picked the highest span non-error design point from the plot which had a 6.1kOhm fixed resistor.

During the interview, P04 compared our workflow to choosing parts in a mainstream board tool's parts library. In particular, P04 noted the importance of not being overwhelmed with choices, especially when those choices aren't meaningfully different from

[11]https://github.com/jankae/LCR

[12]Defined using schematic import [15] instead of in HDL, but parts like resistors and capacitors map down to HDL parts with the parts-table-based design space

[13]http://people.ece.cornell.edu/land/courses/ece4760/FinalProjects/f2017/nas256_jbc262/nas256_jbc262/website/index.html

	Project	Design Space Explored	PCB Experience
P01	High voltage sensor	Voltage regulator, resistive divider parts	11 years (industry)
P02	LCR meter	Voltage regulators, LCR frontend parts	4 years (student)
P03	Greenhouse Sensor	Microcontroller, voltage regulators	1.5 years (student)
P04	Sleep Tracker	Resistive divider parts	< 1 year (student)
P05	Mechanical Keyboard	Microcontroller, switch matrix diodes	minimal (student)
P06	Game Controller	Microcontroller, voltage regulator	minimal (student)

* Experience defined as total years, not years in current role

Table 2: Summary of study participants.

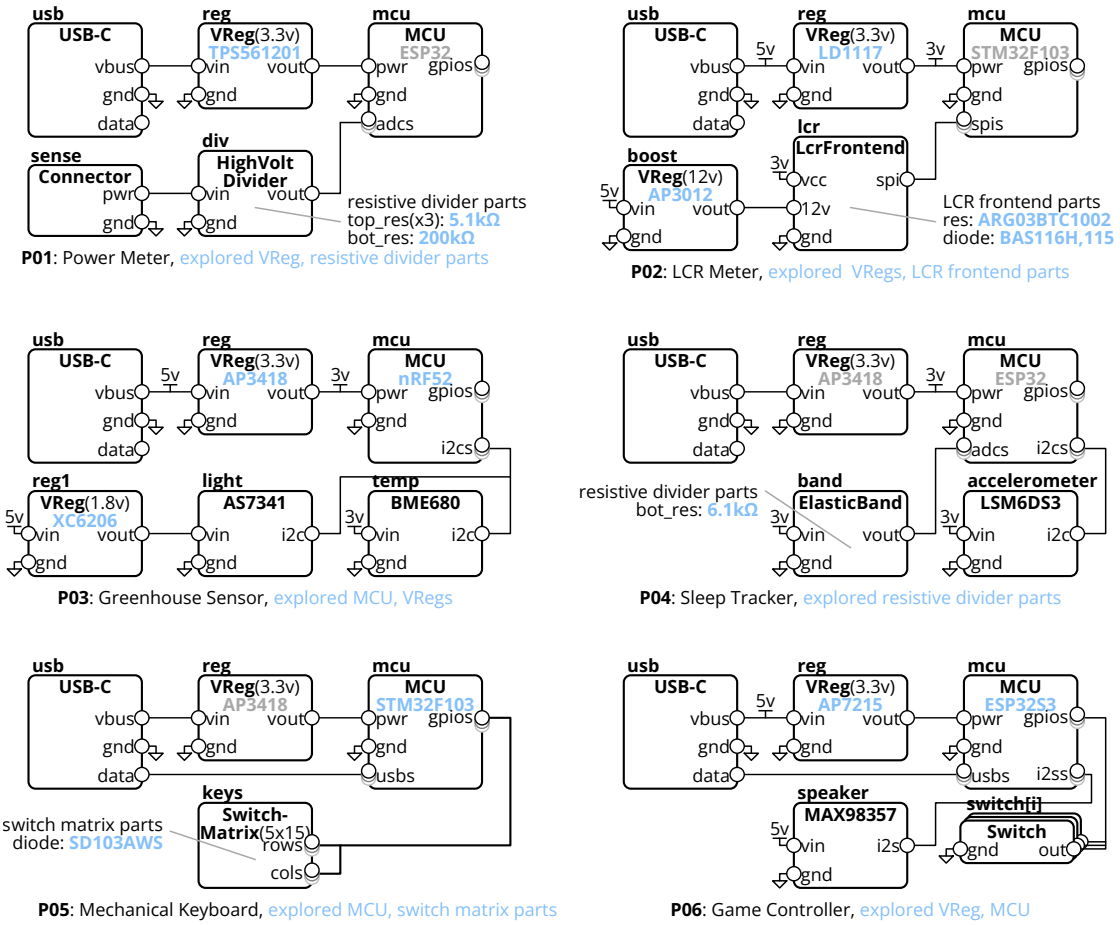

Figure 5: Simplified block diagrams of participant designs. Design choices selected using DSE indicated in blue, manually-selected subclass choices or choices carried over from the tutorial indicated in grey. Some components not related to DSE omitted for brevity.

the substantial list of parts in mainstream tools. While part of the simplicity of our tool came from the limited parts library, another part was the use of parameterized abstract parts (e.g., resistor) and the use of reasonable defaults to avoid requiring the user to make excessive choices.

6.1.5 P05: Mechanical Keyboard. P05 designed a mechanical keyboard, making use of the standard library's parameterized (by number of rows and columns) switch matrix generator. Although P05 had not built a PCB before, P05 was a keyboard enthusiast and understood the major parts of a keyboard PCB to put together the high-level design HDL with a little help. Starting from the tutorial, P05 added a 5x15 switch matrix, which exceeded the available microcontroller pins and caused an error. While P05 initially tried manually reducing the switch matrix size until it stopped giving errors, we suggested trying alternative microcontrollers. P05 used the DSE tool to list the compatible microcontrollers and by area and component count, then (partially on our advice) cross-referenced QMK's (open-source keyboard firmware) supported microcontroller

list to narrow it down to two choices. Further examining the smaller (by area) RP2040-based design, P05 found it had a red proven indicator and finally selected the last, STM32F103-based design.

We also suggested searching the diode options by prompting P05 to compare the diode size against the boards P05 had. P05 added price and area objective functions, and while initially struggling with setting up and interpreting the scatterplot, eventually chose the cheapest and smallest diode.

6.1.6 P06: Game Controller. P06's project idea started with a game controller, though the scope expanded after browsing the library to also include an I2S (digital) speaker driver and more. P06 started with an empty design, though quickly re-created the basic USB connector, voltage regulator, and microcontroller design from the tutorial. P06 then inserted application-specific blocks from the library like switches and displays, then connected ports until the compiler errors went away.

With a completed design, P06 then used DSE to resolve the voltage regulator and microcontroller choice. For the voltage regulator, while P06 initially considered a switching converter given its higher efficiency, we discussed the associated layout challenges of a high-frequency switching circuit for their first board, prompting P06 to instead choose the simpler linear regulator.

6.2 Trends and Patterns

We noticed several trends across participants' use of DSE. This section focuses on observations, while takeaways and ideas are in the discussion in Section 7.

6.2.1 Validity. Overall, participants relied on the validity indicator to avoid choices that wouldn't work, e.g., microcontrollers without requested number of GPIOs. P05 in particular noted that this was helpful for beginners to push out a working board with minimal knowledge:

> Clean slate, like I don't know anything. So if I was to say, I want to design a PCB. I feel like I can utilize this program and its features and tools and it will push something out.

The other more novice participants also mentioned the usefulness of the validity indicator, including the level of integration of component data into a usable interface.

6.2.2 Objective Functions. Of the objective functions provided, participants generally used the cost, component count, area, and current draw to select between feasible points. As these points were all introduced in the tutorial, it is unclear if these are the most common optimization criteria for PCB design in general, or merely what was simplest for participants.

Outside the modeling of our system, programming considerations for microcontrollers were also important. For instance, P05 looked for a microcontroller with QMK firmware support, and P06 checked to make sure that the selected microcontroller was compatible with Arduino. P02 noted the potentially significant downstream impacts of microcontroller choice:

> Generally I find the software side a lot more frustrating, it takes me longer especially if there's some weird quirk, and there's not much documentation [...] I think I spent a whole day on how to get the DAC on the STM32 chip to work.

Participants also mentioned other considerations, for example P03 noted the general reputation of nRF52 microcontrollers being more power efficient than ESP32 microcontrollers and P02 discussed the difficulty of soldering smaller parts.

6.2.3 Search Limitations. Not all search operations that participants wanted were supported by the system. Some of these search operations would be conceptually straightforward if the libraries were expanded, like detent count for encoders. Others might be intuitively reasonable but not possible given the compatible components definition which requires identical electrical interfaces, like LCDs which have diverse interfaces.

6.2.4 Proven Data. While participants didn't plot the proven objective function outside the tutorial, some made use of the proven counts in the parts library. P04 noted similarities to a popularity indicator in a mainstream tool's library, potentially useful as a tiebreaker to prefer the part with a more active community. Similarly, when confronted with a list of several displays, P06 picked the only one marked as being built prior.

More broadly, P01 discussed a preference for familiar parts:

> At some point I have to pick it the first time, which is to say, I like this MCU [microcontroller], let me try it out. Oh, after using it, this is very good MCU. Let me use it for the rest of my life or something like that.

Furthermore, P01 discussed multiple levels of preferred components: whether an they've used the part, whether the organization they're part of has used it, and if lots of people in general use the part.

6.2.5 Performance. As participants swept through wildly different design spaces of different complexities, each had different experiences with the system performance: simpler sweeps over a few resistor combinations take seconds, while sweeping over all of almost 400 diodes takes about three minutes. Participants' backgrounds also factored into what they felt was normal, for example, P05 was used to 10-minute Excel scripts.

Overall, most participants found performance to be acceptable for interactive design space exploration, though some also expanded on it further. For example, P01 commented on the trade-offs of fast results over complete results:

> I don't think I want 10,000 parts in five minutes. I think I want like 200 parts in 20 seconds. And then, and then I can be like, okay, which one of these points has whatever number of designs, and then go from there.

P02 also suggested a pre-filtering interface (like the parametric part selection tools from major component distributors like Digi-Key and Mouser) instead of testing everything in what may be a substantial list of matching parts:

> I think a lot of times you have a general idea of what value you want, so it wouldn't make sense to go through all of them.

Uniquely, P03 had a design space of two voltage regulators and a microcontroller, and was setting up the combinational search

(thousands of points) when we told them that it would be computationally prohibitive. P03 then ran the sweeps sequentially. P06 also remarked that the benefits of sequential searches may not be obvious:

> I think normal people will try to use a combined, because it's more trivial, right? [...] I wasn't really thinking, oh, I can run them separately to get a better timing.

6.2.6 Plots and Tables. While all participants made use of the scatterplot, only P01 and P06 used the parallel coordinates plots outside the guided tutorial. Part of the reason may be that most only optimized for one or two objectives, while P01 was interested in a combination of output voltage range, signal strength, and price. P06 noted the data density benefits:

> I can see more data in one screen.

Both P01 and P04 also noted the usefulness of the table view and its high-precision numeric format, especially when points may be close together on the plots. P05 noted a personal preference for tables instead of graphs.

6.2.7 Novices vs. Experts. While the more expert participants P01 and P02 looked into optimizing parts (including overriding a default selection that wasn't causing errors) and exploring the design space on their own initiative, the more novice participants P03, P04, P05, P06 ran searches to resolve errors. However, with a little explanation of the underlying circuit and design considerations from us, all participants were able to use the DSE tool to find a working solution even where they may not have sufficiently understood things to come up with a solution independently of the tool.

By sheer luck, both P01's and P04's design space exploration focused on a resistive divider circuit and makes for an interesting comparison case across skill levels. The more experienced P01 immediately knew to optimize for output range and signal strength and defined a search space of some E6 resistor values, as would be best practice. With a few promising results on a coarse grid, P01 then refined the search space with a finer grid. On the other hand, the more novice P04 only started design space exploration when prompted by an error on the divider and largely used the DSE tool to effectively guess-and-check solutions by scanning around the default resistance in arbitrary 5kOhm increments. P04 also needed prompting to think about the effect of signal range as an important factor and optimization objective and ultimately settled on the best signal range result with the coarse 5kOhm search grid instead of looking to optimize further.

7 DISCUSSION AND FUTURE WORK

Our observations and the user feedback suggest both design considerations for future tools and potential areas for future investigation.

7.1 Main Takeaways

Overall, our takeaway from the user study is that this method of user-guided design space exploration broadly works and helps a wide range of users achieve their goals. For novices, DSE helps them resolve lower-level technical details to get to a valid design, especially taking into account the requirements of their particular application (e.g., IOs needed on a microcontroller, output voltage on a regulator) instead of generalized advice. For more advanced users, DSE can help users assess trade-offs and choose not just a valid design, but an optimized design. Compared to other recent work in Table 1, we believe this provides a unique combination of lower floor and accessibility to novices while maintaining a high ceiling and usefulness for experts.

Crucially, over automated approaches, DSE also allows the user to bring their own expertise to the table and provides an escape hatch of sorts to take into account requirements outside the modelling of the system. We observed this during the user study across both expert and novice users, such as the various factors driving the choice of microcontroller, or the preference for popular parts.

Furthermore, as some participants primarily started searches when prompted by an error or by us, DSE may be an effective technique for error recovery. Future work could look at interfaces and techniques to go from an error to DSE seamlessly.

7.2 Towards Practical Usage

Our hope is that the techniques and systems we develop could be part of the next generation of board design tools. Yet, more is needed beyond what is feasible from an academic proof-of-concept.

Like many electronics tools presented in recent work, ours, too, is library-based and requires a library of parts with detailed modeling – including electrical limitations and well beyond the simple pin, symbol, and footprint used in mainstream schematic tools. While these libraries encapsulate substantial knowledge and can be shared and re-used, they also have a higher upfront construction cost – in our experience, from tens of minutes for simpler chips like op-amps, to an hour or more for more complex devices like microcontrollers. Datasheet parsing [43] may allow libraries to scale for simpler chips, but often the datasheet must be understood holistically and more magical automation may bring trust issues [16].

More practically, as an open-source system, a community may contribute libraries. However, there may be a chicken-and-egg problem: we expect that a sufficient set of libraries is necessary for adoption (as for programming languages in general [24]), yet adoption is necessary for a community to start contributing. Component manufacturers may also provide libraries for their parts - effectively providing an easy-to-use, machine-executable datasheet that directly integrates into a board design tool - but we expect that this too will depend on adoption.

In practice, though, the existing libraries were mostly sufficient for the wide set of examples in the user study. So, one larger question may be, how much is good enough?

7.3 Scaling and Performance

Performance-wise, a full design space sweep optimized with partial compilation seemed acceptable to participants for interactive DSE. While our design space is tiny compared to the space of all electronics parts, it does cover meaningful choices and allowed participants to complete their designs. Furthermore, curated libraries of common components do exist [34] with roughly similar numbers of microcontrollers and voltage regulators as our libraries. It may be that smaller, focused libraries for specific purposes are ultimately better, both psychologically helping avoid choice overload (as mentioned by participants), and in reducing performance impact.

Where a large design space is needed, pre-filtering may be useful, such as the user-driven interface P02 suggested based on the parametric search interface of component distributors. Alternatively, heuristics to filter out highly-suboptimal areas of the search space could be baked into the search tool itself. Probabilistic filters (where some, but not all, expected unproductive choices are removed) may also be a reasonable compromise that still helps users understand the larger design space but waste less time.

Ultimately, as the user guides the DSE process, they are responsible for keeping the search space reasonable, such as through decomposing a complex design into several, independent search operations. Future work may explore proactively providing automated guidance (especially for novices), such as identifying where the design space can be decoupled.

7.4 Discoverability

Understanding the degrees of freedom and objective functions was a stumbling block for all participants. This mostly relates to the parameters within the electronics model (e.g., current draw on a pin), which the underlying HDL only understands as a variable and our tool does not provide any specialized graphical affordances for (in contrast to e.g., the LTSPICE circuit simulator [5], where mousing over a pin prompts the user to view the current into the pin and mousing over a wire prompts the user to view the voltage on the wire).

Some of the above may be addressable with better interfaces, such as indicators of which parameters are inputs or outputs, potentially also using static analysis techniques to tell the user the impacts of sweeping some variable. Methods to highlight common design parameters and objective functions may also help, perhaps aided by keyword search and some intelligent ranking of results. Alternatively, perhaps the simple (but boring) answer is just better documentation.

In general, this may be an under-explored area for DSE tools. Prior work in mechanical DSE has a few well-known and intuitive objective functions like mass [30], while more advanced tools like OpenMETA [36] are geared towards experts with less of an emphasis on the learning curve. In our case, we had a tool structured like OpenMETA (general parameter system), with an included electronics framework and library, and aiming to be useful to novices and experts – the tension of maintaining a low floor while enabling a high ceiling and wide walls.

7.5 Better Supporting Novices

While DSE helped all our participants get to a board, they still needed to know enough about electronics to design the system architecture – though not a problem for our participants, this may not be true for even more novice users. An even higher-level design abstraction, like the functional model in Trigger-Action-Circuits [3], might help. Higher-level design models may also expand the search beyond mere electrical compatibility and towards functionality, potentially allowing searches over the aforementioned LCDs that do not have identical electrical interfaces.

However, this starts getting into synthesis territory and raises issues of less interpretable interfaces, compared to our relatively straightforward "all satisfying microcontrollers". Future work might examine hybrid solutions like synthesis or even mixed-initiative circuit autocomplete [18] to get to a system architecture followed by optional DSE to refine the part selections.

Additionally, when working with lower level subcircuits like resistive dividers, participants also need to know enough about its underlying physics to set up the search space, as P04 initially struggled with. Some default choices were also easy to overlook, like for P05's excessively large keyboard matrix diodes. For these, more domain knowledge might be encoded in the libraries to either provide a better default or help set up a search space.

7.6 Study Limitations

As for our user study, we frame our work as more initial and exploratory, focusing on generating ideas for future work and examining the limitations of our prior understanding. We believe we have obtained in-depth data in the realistic setting of participant-defined projects - for example, our results on the wide variety of factors in part choice can emphasize the need for flexibility in tools. However, we have traded off sample size and standardized procedures, both of which limit generalizability. Future work could run larger and more controlled studies, trading off realism for being able to quantitatively measure the effectiveness of DSE compared to either mainstream schematic capture or high-level design.

8 CONCLUSION

Although much recent work on electronics focuses on either novice-friendly but limiting fully automated design or higher-learning-curve but flexible manual design, here we explore a middle point of user-guided design space exploration in a library-based system where the tool helps users understand and choose between parts in the context of their particular design. Feedback from a realistic and in-depth user study where participants of wide skill levels completed their projects in our system suggests that this is a promising approach with a wide variety of applications including optimization and error recovery, along with raising ideas for future work.

In the larger picture, we hope the ideas here can apply beyond electronics to other fields of design and is another step towards powerful design tools for everyone.

ACKNOWLEDGMENTS

This work was supported in part by DARPA grants HR00112110008 and FA8750-20-C-0156 (SDCPS). The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof, nor does this imply any official endorsement. Approved for public release; distribution is unlimited.

REFERENCES

[1] 2003. *Design Space Visualization and Its Application to a Design by Shopping Paradigm.* International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Vol. Volume 2: 29th Design Automation Conference, Parts A and B. https://doi.org/10.1115/DETC2003/DAC-48785 arXiv:https://asmedigitalcollection.asme.org/IDETC-CIE/proceedings-pdf/IDETC-CIE2003/37009/795/2594167/795_1.pdf

[2] Audrey Abi Akle, Stéphanie Minel, and Bernard Yannou. 2017. Information visualization for selection in Design by Shopping. *Research in Engineering Design* 28 (2017), 99–117. https://doi.org/10.1007/s00163-016-0235-2

[3] Fraser Anderson, Tovi Grossman, and George Fitzmaurice. 2017. 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* (Québec City, QC, Canada) *(UIST '17)*. ACM, New York, NY, USA, 331–342. https://doi.org/10.1145/3126594.3126637

[4] circuito.io. 2020. *Circuit Design App for Makers- circuito.io.* https://www.circuito.io/

[5] Analog Devices. 1999. *LTSpice.* https://www.analog.com/en/design-center/design-tools-and-calculators/ltspice-simulator.html

[6] Daniel Drew, Julie L Newcomb, William McGrath, Filip Maksimovic, David Mellis, and Björn Hartmann. 2016. The toastboard: Ubiquitous instrumentation and automated checking of breadboarded circuits. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. 677–686.

[7] EDASolver. 2016. *EDASolver: Welcome to Functional EDA.* https://edasolver.com

[8] Jorge Garza, Devon J. Merrill, and Steven Swanson. 2021. Appliancizer: Transforming Web Pages into Electronic Devices. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 415, 13 pages. https://doi.org/10.1145/3411764.3445732

[9] Nicholas Jennings, Ananya Nandy, Xinyi Zhu, Yuting Wang, Fanping Sui, James Smith, and Bjoern Hartmann. 2022. GeneratiVR: Spatial Interactions in Virtual Reality to Explore Generative Design Spaces. In *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI EA '22)*. Association for Computing Machinery, New York, NY, USA, Article 366, 6 pages. https://doi.org/10.1145/3491101.3519616

[10] Wenhao Jia, Kelly A. Shaw, and Margaret Martonosi. 2012. Stargazer: Automated regression-based GPU design space exploration. In *2012 IEEE International Symposium on Performance Analysis of Systems & Software*. 2–13. https://doi.org/10.1109/ISPASS.2012.6189201

[11] Rushil Khurana and Steve Hodges. 2020. Beyond the Prototype: Understanding the Challenge of Scaling Hardware Device Production. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–11.

[12] André Knörig, Reto Wettach, and Jonathan Cohen. 2009. Fritzing: A Tool for Advancing Electronic Prototyping for Designers. In *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction* (Cambridge, United Kingdom) *(TEI '09)*. ACM, New York, NY, USA, 351–358. https://doi.org/10.1145/1517664.1517735

[13] Richard Lin, Rohit Ramesh, Connie Chi, Nikhil Jain, Ryan Nuqui, Prabal Dutta, and Björn Hartmann. 2020. Polymorphic Blocks: Unifying High-Level Specification and Low-Level Control for Circuit Board Design. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '20)*. Association for Computing Machinery, New York, NY, USA, 529–540. https://doi.org/10.1145/3379337.3415860

[14] Richard Lin, Rohit Ramesh, Prabal Dutta, Bjoern Hartmann, and Ankur Mehta. 2022. Computational Support for Multiplicity in Hierarchical Electronics Design. In *Proceedings of the 7th Annual ACM Symposium on Computational Fabrication* (Seattle, WA, USA) *(SCF '22)*. Association for Computing Machinery, New York, NY, USA, Article 1, 11 pages. https://doi.org/10.1145/3559400.3561997

[15] Richard Lin, Rohit Ramesh, Prabal Dutta, Bjoern Hartmann, and Ankur Mehta. 2023. Greyboxing: towards domain-specific representations for domain-specific languages in electronics design. In *PLATEAU Workshop* (Pittsburgh, PA, USA) *(PLATEAU 2023)*. https://doi.org/10.1184/R1/22277344.v1

[16] Richard Lin, Rohit Ramesh, Antonio Iannopollo, Alberto Sangiovanni Vincentelli, Prabal Dutta, Elad Alon, and Björn Hartmann. 2019. Beyond Schematic Capture: Meaningful Abstractions for Better Electronics Design Tools. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, Article 283, 13 pages. https://doi.org/10.1145/3290605.3300513

[17] Richard Lin, Rohit Ramesh, Nikhil Jain, Josephine Koe, Ryan Nuqui, Prabal Dutta, and Bjoern Hartmann. 2021. Weaving Schematics and Code: Interactive Visual Editing for Hardware Description Languages. In *The 34th Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '21)*. Association for Computing Machinery, New York, NY, USA, 1039–1049. https://doi.org/10.1145/3472749.3474804

[18] Jo-Yu Lo, Da-Yuan Huang, Tzu-Sheng Kuo, Chen-Kuo Sun, Jun Gong, Teddy Seyed, Xing-Dong Yang, and Bing-Yu Chen. 2019. AutoFritz: Autocomplete for Prototyping Virtual Breadboard Circuits. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, Article 403, 13 pages. https://doi.org/10.1145/3290605.3300633

[19] Aran Lunzer and Kasper Hornbæk. 2008. Subjunctive Interfaces: Extending Applications to Support Parallel Setup, Viewing and Control of Alternative Scenarios. *ACM Trans. Comput.-Hum. Interact.* 14, 4, Article 17 (jan 2008), 44 pages. https://doi.org/10.1145/1314683.1314685

[20] Justin Matejka, Michael Glueck, Erin Bradner, Ali Hashemi, Tovi Grossman, and George Fitzmaurice. 2018. Dream Lens: Exploration and Visualization of Large-Scale Generative Design Datasets. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3173574.3173943

[21] David A. Mellis, Leah Buechley, Mitchel Resnick, and Björn Hartmann. 2016. Engaging Amateurs in the Design, Fabrication, and Assembly of Electronic Devices. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems* (Brisbane, QLD, Australia) *(DIS '16)*. Association for Computing Machinery, New York, NY, USA, 1270–1281. https://doi.org/10.1145/2901790.2901833

[22] Devon J. Merrill, Jorge Garza, and Steven Swanson. 2019. Echidna: Mixed-Domain Computational Implementation via Decision Trees. In *Proceedings of the ACM Symposium on Computational Fabrication* (Pittsburgh, Pennsylvania) *(SCF '19)*. Association for Computing Machinery, New York, NY, USA, Article 5, 12 pages. https://doi.org/10.1145/3328939.3329004

[23] Devon J. Merrill and Steven Swanson. 2019. Reducing Instructor Workload in an Introductory Robotics Course via Computational Design. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (Minneapolis, MN, USA) *(SIGCSE '19)*. Association for Computing Machinery, New York, NY, USA, 592–598. https://doi.org/10.1145/3287324.3287506

[24] Leo A. Meyerovich and Ariel S. Rabkin. 2013. Empirical Analysis of Programming Language Adoption. *SIGPLAN Not.* 48, 10 (oct 2013), 1–18. https://doi.org/10.1145/2544173.2509515

[25] OpenMETA. 2020. *LED Tutorial.* http://docs.metamorphsoftware.com/doc/tutorials/led_tutorial/led_tutorial.html

[26] Andy D. Pimentel. 2017. Exploring Exploration: A Tutorial Introduction to Embedded Systems Design Space Exploration. *IEEE Design & Test* 34, 1 (2017), 77–90. https://doi.org/10.1109/MDAT.2016.2626445

[27] Rohit Ramesh, Richard Lin, Antonio Iannopollo, Alberto Sangiovanni-Vincentelli, Björn Hartmann, and Prabal Dutta. 2017. Turning Coders into Makers: The Promise of Embedded Design Generation. In *Proceedings of the 1st Annual ACM Symposium on Computational Fabrication* (Cambridge, Massachusetts) *(SCF '17)*. ACM, New York, NY, USA, Article 4, 10 pages. https://doi.org/10.1145/3083157.3083159

[28] Benjamin Carrion Schafer and Zi Wang. 2020. High-Level Synthesis Design Space Exploration: Past, Present, and Future. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39, 10 (2020), 2628–2639. https://doi.org/10.1109/TCAD.2019.2943570

[29] Adriana Schulz, Harrison Wang, Eitan Grinspun, Justin Solomon, and Wojciech Matusik. 2018. Interactive Exploration of Design Trade-Offs. *ACM Trans. Graph.* 37, 4, Article 131 (jul 2018), 14 pages. https://doi.org/10.1145/3197517.3201385

[30] Adriana Schulz, Jie Xu, Bo Zhu, Changxi Zheng, Eitan Grinspun, and Wojciech Matusik. 2017. Interactive Design Space Exploration and Optimization for CAD Models. *ACM Trans. Graph.* 36, 4, Article 157 (jul 2017), 14 pages. https://doi.org/10.1145/3072959.3073688

[31] Ben Shneiderman. 2002. Creativity Support Tools. *Commun. ACM* 45, 10 (Oct. 2002), 116–120. https://doi.org/10.1145/570907.570945

[32] Ben Shneiderman. 2007. Creativity Support Tools: Accelerating Discovery and Innovation. *Commun. ACM* 50, 12 (Dec. 2007), 20–32. https://doi.org/10.1145/1323688.1323689

[33] Ben Shneiderman. 2009. Creativity support tools: A grand challenge for HCI researchers. *Engineering the User Interface* (2009), 1–9.

[34] SnapEDA. 2023. *Open Parts Library.* https://www.snapeda.com/libraries/seeedstudio/open-parts-library/

[35] Sparkfun. 2020. *À La Carte.* https://alc.sparkfun.com/

[36] Janos Sztipanovits, Ted Bapty, Sandeep Neema, Larry Howard, and Ethan Jackson. 2014. OpenMETA: A Model- and Component-Based Design Tool Chain for Cyber-Physical Systems. In *From Programs to Systems. The Systems perspective in Computing*, Saddek Bensalem, Yassine Lakhneck, and Axel Legay (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 235–248.

[37] Toktam Taghavi and Andy D. Pimentel. 2010. VMODEX: A visualization tool for multi-objective Design Space Exploration. In *2010 International Conference on Field-Programmable Technology*. 357–360. https://doi.org/10.1109/FPT.2010.5681428

[38] Michael Terry and Elizabeth D. Mynatt. 2002. Recognizing Creative Needs in User Interface Design. In *Proceedings of the 4th Conference on Creativity & Cognition* (Loughborough, UK) *(C&C '02)*. Association for Computing Machinery, New York, NY, USA, 38–44. https://doi.org/10.1145/581710.581718

[39] Upverter. 2023. *Upveter | Modular Electronics Design.* https://modular.upverter.com/

[40] Laurens Van Der Maaten, Eric Postma, Jaap Van den Herik, et al. 2009. Dimensionality Reduction: A Comparative Review. *J Mach Learn Res* 10, 66-71 (2009), 13. https://members.loria.fr/moberger/Enseignement/AVR/Exposes/TR_Dimensiereductie.pdf

[41] Chiuan Wang, Hsuan-Ming Yeh, Bryan Wang, Te-Yen Wu, Hsin-Ruey Tsai, Rong-Hao Liang, Yi-Ping Hung, and Mike Y. Chen. 2016. CircuitStack: Supporting Rapid Prototyping and Evolution of Electronic Circuits. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (Tokyo, Japan) *(UIST '16)*. ACM, New York, NY, USA, 687–695. https://doi.org/10.1145/2984511.2984527

[42] R.S. Weiss. 1995. *Learning From Strangers: The Art and Method of Qualitative Interview Studies.* Free Press. https://books.google.com/books?id=i2RzQbiEiD4C

[43] Sen Wu, Luke Hsiao, Xiao Cheng, Braden Hancock, Theodoros Rekatsinas, Philip Levis, and Christopher Ré. 2018. Fonduer: Knowledge Base Construction from Richly Formatted Data. In *Proceedings of the 2018 International Conference on Management of Data* (Houston, TX, USA) *(SIGMOD '18)*. Association for Computing Machinery, New York, NY, USA, 1301–1316. https://doi.org/10.1145/3183713.3183729

[44] Loutfouz Zaman, Wolfgang Stuerzlinger, Christian Neugebauer, Rob Woodbury, Maher Elkhaldi, Naghmi Shireen, and Michael Terry. 2015. GEM-NI: A System for Creating and Managing Alternatives In Generative Design. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (Seoul, Republic of Korea) *(CHI '15)*. Association for Computing Machinery, New York, NY, USA, 1201–1210. https://doi.org/10.1145/2702123.2702398