Pinpoint: A PCB Debugging Pipeline Using Interruptible Routing and Instrumentation

Evan Strasnick
Stanford University
Stanford, CA
estrasni@stanford.edu

Sean Follmer
Stanford University
Stanford, CA
sfollmer@stanford.edu

Maneesh Agrawala
Stanford University
Stanford, CA
maneesh@cs.stanford.edu

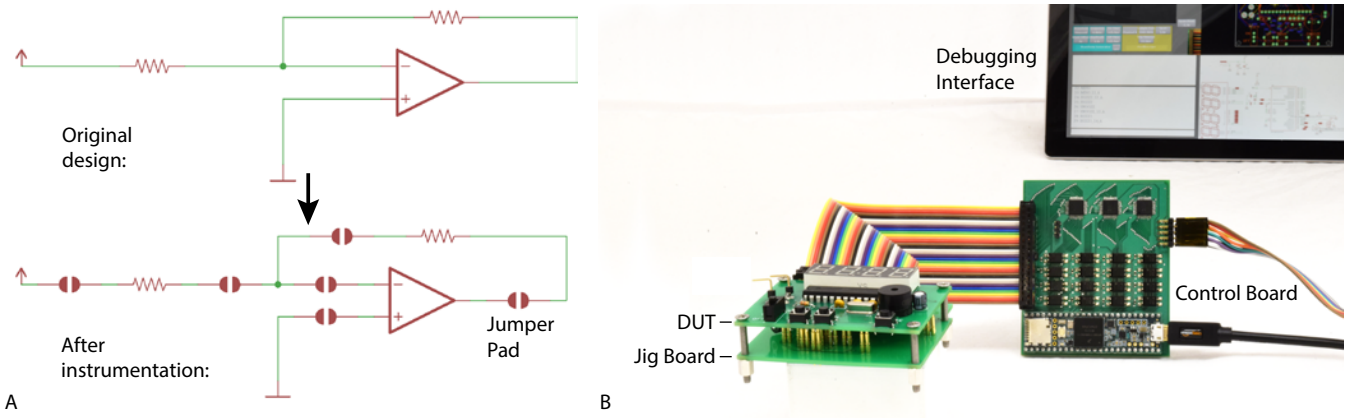

Figure 1: Pinpoint provides designers with new PCB debugging capabilities. (A) First, Pinpoint automatically instruments a circuit design with "jumper pads" that enable programmatic control over each connection. (B) It then produces designs for a jig interface between the user's PCB (the Device Under Test, DUT) and Pinpoint's custom testing hardware (the control board). A graphical interface enables debugging features such as probing, isolating parts of the circuit, and authoring functional tests.

ABSTRACT

Difficulties in accessing, isolating, and iterating on the components and connections of a printed circuit board (PCB) create unique challenges in PCB debugging. Manual probing methods are slow and error prone, and even dedicated PCB testing equipment remains limited by its inability to modify the circuit during testing. We present Pinpoint, a tool that facilitates in-circuit PCB debugging through techniques such as programmatically probing signals, dynamically disconnecting components and subcircuits to test in isolation, and splicing in new elements to explore potential modifications. Pinpoint automatically instruments a PCB design and generates designs for a physical jig board that interfaces the user's PCB to our custom testing hardware and to software tools. We evaluate Pinpoint's ability to facilitate the debugging of various PCB issues by instrumenting and testing different classes of boards, as well as by characterizing its technical limitations and by soliciting feedback through a guided exploration with PCB designers.

CHI 2019, May 4–9, 2019, Glasgow, Scotland Uk

ACM ISBN 978-1-4503-5970-2/19/05…$15.00
https://doi.org/10.1145/3290605.3300278

CCS CONCEPTS

• **Human-centered computing → Human computer interaction (HCI)**; • **Hardware → Bug detection, localization and diagnosis**; *PCB design and layout*;

KEYWORDS

printed circuit board, PCB, debugging, design tool, jig, in-circuit testing, design-for-test, instrumentation

ACM Reference Format:
Evan Strasnick, Sean Follmer, and Maneesh Agrawala. 2019. Pinpoint: A PCB Debugging Pipeline Using Interruptible Routing and Instrumentation. In *CHI Conference on Human Factors in Computing Systems Proceedings (CHI 2019), May 4–9, 2019, Glasgow, Scotland Uk*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3290605.3300278

1 INTRODUCTION

Printed circuit boards (PCBs) are typically the final realization of a circuit design due to robustness, compactness, and ease of mass production, but pose significant challenges in debugging. Bugs can fall into several classes: *Construction errors* involve a mismatch between the design and the constructed circuit. These include continuity problems (short circuits and open circuits), faulty components, and fabrication defects [12]. *Conceptual errors* result from a mismatch between specification and design – for example, routing components incorrectly or operating them outside of their rated conditions. Errors of *noise* involve external interference with an otherwise correct design. Noise can result from interactions between subcircuits, e.g. drawing current causes fluctuations on a voltage supply, or can enter the circuit from an outside source, e.g. through capacitive or inductive coupling.

Debugging in a PCB context is especially complicated by difficulties in *accessing*, *isolating*, and *iterating* on the elements and connections of the circuit. Construction errors are typically invisible and inaccessible, hidden beneath or within surface-mounted components, and manually probing exposed pins or pads to observe signals is slow and error prone. Testing a fully assembled board ("in-circuit testing") is limited in that many fundamental tests require isolating parts of the circuit to test (e.g. checking component values), but the designer cannot modify the fixed connections on the board. Similarly, after identifying a conceptual issue, the designer typically cannot explore potential solutions or iterate on the design without refabricating the board.

The manual tools commonly used to debug PCBs (oscilloscope, multimeter, etc.) are versatile, but alone offer limited support for *test-oriented* debugging. As in software debugging, functional tests can help to quickly identify unexpected behaviors of the circuit. However, while researchers have developed a powerful ecosystem of tools for software debugging, many PCB designers lack access to an effective *test infrastructure*. Dedicated test platforms – such as a flying-probe or bed-of-nails tester – interface signals on the board with software tools, enabling designers to programmatically define tests that save time, reduce errors, and provide opportunities for automation. However, due to cost, test platforms are typically only used by professional teams for projects at scale, leaving makers, researchers, and independent designers without access to effective debugging methods. More importantly, these testers remain limited by their inability to isolate and modify parts of the circuit during testing. They cannot, for example, remove a component to verify its resistance, explore an alternative configuration, or perform integration testing as subcircuits are combined.

We present Pinpoint, a tool providing designers with a range of new in-circuit PCB testing methods at low-cost. Pinpoint improves *access* by enabling users to quickly visualize and record signals from any instrumented point on the board. It can programmatically connect and disconnect components and subcircuits, enabling the designer to temporarily *isolate* circuit elements, and new elements can be spliced into the circuit to *iterate* and test modifications. Finally, Pinpoint supports a test-oriented workflow through the injection of test signals, automatic detection of common issues, and authoring of functional tests.

Pinpoint enables these interactions via three primary components, shown in Figure 1. It first automatically instruments a PCB design with "jumper pads" (paired test pads joined by a relay connection, Figure 1.A), as well as generates a new "jig board" design that serves as a "bed-of-nails"-style interface to the device under test (DUT). Once connected, Pinpoint's custom testing hardware can probe the DUT, inject signals, isolate components, and splice in new elements (Figure 1.B). Finally, a graphical user interface provides access to test features, such as visualizing signals, authoring unit tests, and automatically scanning for common issues (Figure 5). We evaluate Pinpoint's ability to facilitate the debugging of various PCB issues by instrumenting and testing different classes of boards, as well as by characterizing its technical limitations and by soliciting feedback through a guided exploration with PCB designers.

2 RELATED WORK

State of the art PCB debugging tools automate electrical checks for construction errors ("fault detection") as well as facilitate functional testing [6, 12]. These tests can be performed before placing components ("bare board tests") or after ("in-circuit tests"). Common test equipment includes flying probe testers, which use actuated contacts to serially probe test sites, and bed-of-nails fixtures, which use fixed contacts laid out to mate with exposed test regions on a DUT. Imaging-based methods also enable optical, thermal, and x-ray detection of construction errors [12]. Software-based testing platforms, such as LabVIEW [18], provide sophisticated data acquisition and functional testing interfaces but require significant manual instrumentation.

As manually authoring tests becomes laborious for large circuits, Automated Test Pattern Generation (ATPG) can assist designers by generating checks for anticipated faults [12]. However, ATPG introduces new complexity, typically requiring an abstracted "fault model" for the target circuit. Pinpoint avoids this complexity by generating common tests without a fault model, while providing a high-level authoring interface for user-defined tests.

Other in-circuit testing methods leverage Built-In Self Test (BIST) capabilities designed into components, such as the widely adopted JTAG standard [15]. JTAG enables a "boundary scan" approach, where designers use an added debug

port to configure built-in test cells of an integrated circuit (IC) on the board. JTAG provides the greatest utility to IC manufacturers, while end-user designers must write specific tests for each JTAG-enabled IC on their board to leverage its capabilities. Boundary scan techniques remain limited by the inability to modify the circuit under test.

Many students, independent designers, and hobbyist makers have turned to DIY methods for affordable testing solutions. For example, some have modified CNC tools (such as a 3D printer) to act as a flying probe tester [2]. Prototyping-oriented test platforms like the Digilent Electronics Explorer [10] allow makers to connect sites on a breadboard to tools like oscilloscopes and logic analyzers. Pinpoint draws inspiration from examples of DIY bed-of-nails fixtures. While currently makers manually design these fixtures for each project, Pinpoint automates jig design generation.

Inspired by the spread of maker practices, research communities have identified new computational support needs in electronic design [8, 17]. These emerging tools have primarily targeted the prototyping stage (e.g. the breadboard), where low cost, fast iteration, and hands-on construction offer a novice-friendly platform for tinkering and learning-by-doing [1, 14, 27, 29, 30]. Many of these systems share motivations with Pinpoint, such as the Toastboard [11], which instruments a breadboard to facilitate signal visualization and functional testing, Bifröst [16], which links electrical events to code execution, and CurrentViz [31], which augments the breadboard with in-circuit current detection. However, while many bugs can be found in the breadboard stage, new issues surface in the transition to the PCB (e.g construction errors, misrouting of components), and the PCB's added difficulties in access, isolation, and iteration create additional barriers to debugging that demand research attention.

Other tools have explored new debugging techniques beyond the prototyping stage of hardware design. For example, BoardLab reduces the designer's cognitive load when probing a PCB by highlighting the probe's location within the design file [13]. Other systems improve debugging by facilitating iteration, either through rapid fabrication techniques [7] or by leveraging interactively reconfigurable hardware [25]. Pinpoint intervenes in each of the design, fabrication, and debugging stages to more holistically address these and other core difficulties.

3 EXAMPLE DEBUGGING SESSION

We present an example PCB design workflow to illustrate the use of Pinpoint in debugging common issues: A student, Elle, is designing a handheld game controller, consisting of a potentiometer-based thumbstick that reports position along a horizontal axis and a capacitive button that detects a finger press (Figure 2).

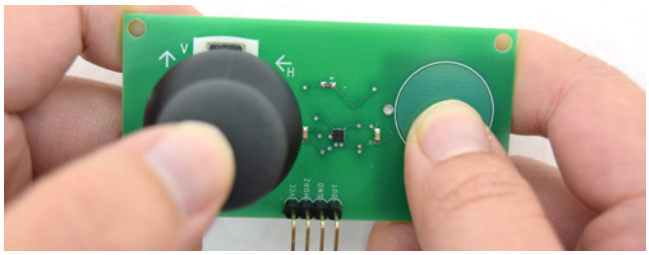

Figure 2: A game controller PCB designed using Pinpoint, featuring a thumbstick and capacitive button.

Board and Jig Creation

Elle uses a circuit design tool to create a schematic then to lay out and route components on a board. Before sending her design for fabrication, she runs Pinpoint's instrumentation script, which automatically splices jumper pads (paired test pads) into the signals on her schematic (Figure 3.A) and board (Figure 3.B). The pads lie on the underside of the board, and divide each signal in two so that Pinpoint can dynamically open and close connections.

Elle fabricates her board, along with a second design generated by Pinpoint called the "jig board" (Figure 3.D). The jig board interfaces her instrumented PCB (the device under test, or DUT) with Pinpoint's custom testing hardware (the "control board", Figure 1.B). Elle assembles the jig board by soldering pogo pins into the provided through-holes, mounts it onto the DUT with screws (Figure 3.E), then connects the control board to the jig board and to her PC.

Debugging with Pinpoint

Probing Signals. When trying out her controller, Elle notices a problem with the thumbstick, which constantly reports a movement to the right. She first uses the built-in oscilloscope in Pinpoint's debugging interface (Figure 5) to probe the output of the thumbstick, finding that it reads at 0V. She can set Pinpoint's two virtual probes on any instrumented signals by clicking on their diagrammatic representations in the schematic, so she does not need to track which pin on her board carries which signal.

Automatic Detection of Construction Errors. Wondering if the error lies in her construction of the board, Elle then uses Pinpoint to automatically scan her board for potential continuity issues. Pinpoint automates these routine checks, checking all pairs of signals for unintended connections. These tests indicate that there may be a short circuit between the output of the thumbstick and the adjacent ground pin. After removing a solder bridge hidden beneath the component, the thumbstick now responds to her inputs.

Subcircuit Isolation. While the thumbstick seems responsive, Elle next notices that it occasionally makes small inputs

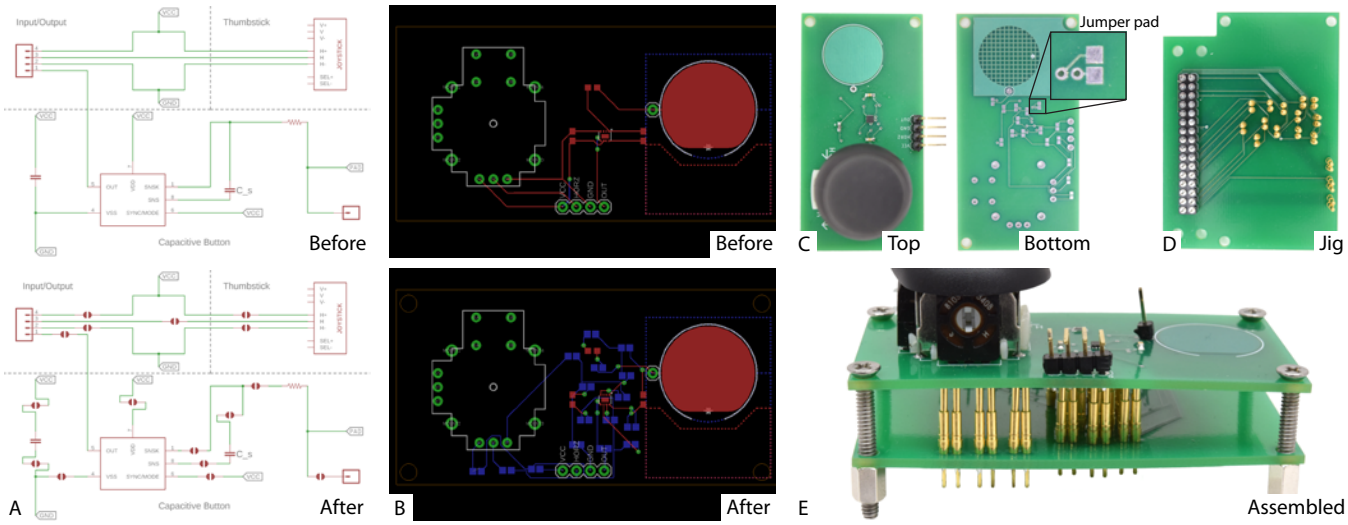

Figure 3: Game controller schematic (A) and board (B) before (top) and after (bottom) instrumentation on all signals. (C) The top and bottom (mirrored for clarity) of the instrumented game controller PCB with added jumper pads (callout). (D) The corresponding jig board generated by Pinpoint. (E) The jig board mates to the DUT with each pogo pin probing a pad.

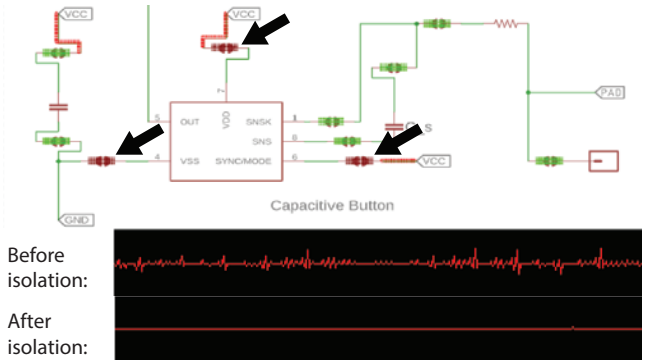

Figure 4: Pinpoint uses interruptible jumper pads to isolate parts of the circuit, e.g. to localize issues. Noise on the circuit's VCC line disappears when opening the indicated VCC and GND connections to the capacitive sensing component.

in random directions. She probes the input voltage (VCC), and finds significant noise on the supply. Hoping to localize the source, Elle temporarily isolates parts of the circuit without desoldering, by instead clicking on jumper pads to open and close connections on her board (Figure 4). When she disconnects the capacitive sensor from the power line, the noise disappears and she recognizes the sensor as the source of the noise. She then updates her design to include an additional decoupling capacitor.

Functional Testing and Test Signal Injection. Elle has also noticed that her capacitive button does not respond to her touch. She loads a collection of unit tests saved from the last circuit she designed with the same sensor – some tests that she authored herself and some imported from component libraries. These tests leverage probing, subcircuit isolation, and signal injection to check properties of a signal (e.g. voltage, frequency, period) or of a board (e.g. continuity between two points). She imports and runs an in-circuit functional test designed by the component manufacturer to verify whether the sensor is properly configured and in working condition. The test isolates the sensing circuit, injects a 3.3V signal at the output of the component, and records to observe characteristic peaks in the response. The test reports no issue, indicating that Elle's sensor is functional.

Circuit Splicing. Elle then decides to try adjusting the sensitivity of her circuit, which is determined by the value of a capacitor, C_s, on the board. She temporarily disconnects it by clicking on the surrounding jumper pads to interrupt the connections at those points. She then splices in a breadboarded circuit – where she can quickly place different capacitors – by connecting jumper wires to highlighted ports on Pinpoint's control board (Figure 6). She experiments with different capacitors on the breadboard and finds that .1μF works better in her circuit than the manufacturer suggested value of 10nF. She additionally splices in a decoupling capacitor over the power and ground connections of the sensing IC to verify that it mitigates the noise she identified earlier.

Finally, having verified the solutions to her current bugs, Elle designs a revised version of her game controller. Once assembled, she repeats the unit tests from before to quickly verify the board's construction. With debugging finished, she then ties the two ends of each jumper pad together with a bit of solder to leave her board in a connected state.

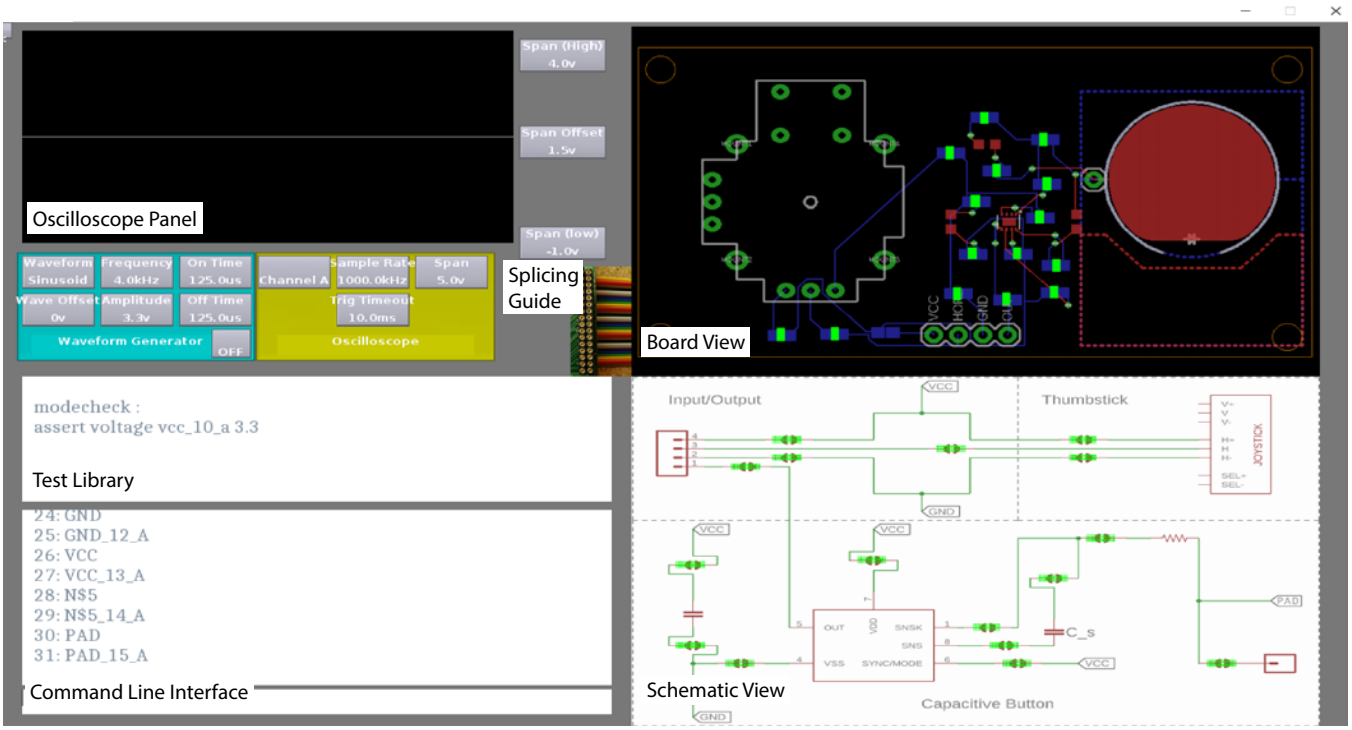

Figure 5: Pinpoint's debugging interface. An oscilloscope view (top left) probes and injects signals. Users author and run tests via the command line below. On the right, the circuit is shown in linked board and schematic representations. Signals clicked in one representation will highlight in the other, as well as on a visual indication of corresponding ports to splice in circuits.

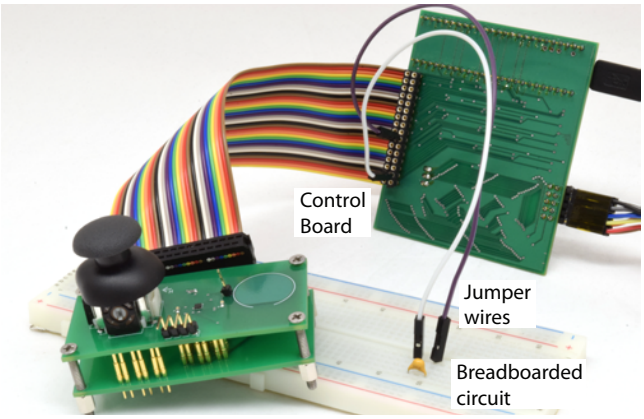

Figure 6: Pinpoint enables designers to splice external circuits into instrumented sites on the DUT. Here, the designer splices in a breadboard to rapidly explore capacitor values.

With respect to the previously identified challenges in PCB debugging, Pinpoint improved Elle's *access* via programmatic signal probing, enabled *isolation* of components to perform in-circuit functional tests, and facilitated *iteration* through spliced-in circuit modifications.

4 IMPLEMENTATION

Pinpoint includes an instrumentation algorithm used during circuit design; fabrication techniques to create a test setup during circuit construction; custom testing hardware ("control board") that probes sites, injects signals, and interrupts connections; and a debugging interface used in testing. We adopted several design constraints, informed by our own experiences and by formative interviews conducted with PCB designers: low cost, rapid assembly, minimal impact on the designer's workflow, and few added board requirements (e.g. required free space, changes to original layout).

Circuit Design Instrumentation

Pinpoint's instrumentation stage is written as a User Language Program for the EAGLE circuit design tool [4]. It automatically adds interruptible connections as jumper pads across all signals in the designer's board (up to a maximum (16) supported by the hardware). Designers can instead specify which signals to instrument, as well as whether the connections should be normally-open (i.e. they solder the pads together when finished debugging) or normally-closed (i.e. they sever the connections as needed for a debugging session).

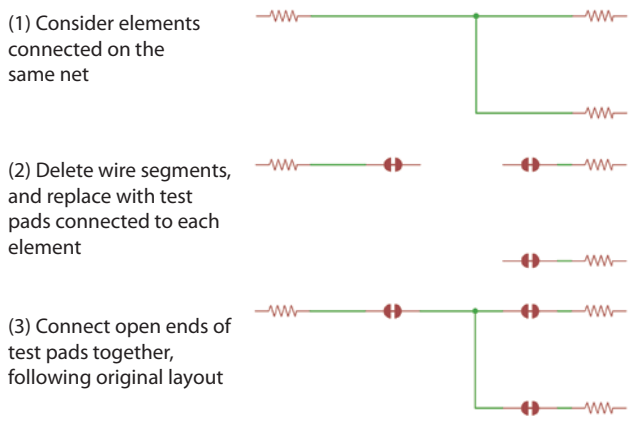

Figure 7: Pinpoint introduces jumper pads with interruptible connections to enable the isolation of each element on a given net. Each pad connects on one end to a circuit element, and the remaining open ends are tied together.

Figure 8: A net with two components has only two states: connected or disconnected. Pinpoint needs only a single test pad to instrument nets in this configuration.

(1) **Splice jumper pads into nets on the schematic.** Pinpoint places jumper pads so as to enable any combination of connections to each *element* (component or subcircuit) on a given *net* (electrical connection shared by a set of components). We satisfy this requirement with a specific topology: each jumper pad connects on one end to a single element, and the other ends of all pads on the net are connected together (Figure 7). We modify the schematic as follows:

```
for net S in the original design do
    for element E on S do
        for wire segment W in S and incident on E do
            Add a jumper pad P at the midpoint of W;
            Delete W;
            Connect one end of P to E on a new net S_E;
        end
    end
    Connect the open ends of all jumper pads together
      on the net S, following deleted segment paths to
      preserve the original appearance of the schematic;
end
```

Thus, a net with n elements requires n jumper pads for full instrumentation. However, as shown in Figure 8, the case where $n = 2$ requires only a single pad to either connect or disconnect the components. We reduce the total number of added pads by using a single splice in these cases.

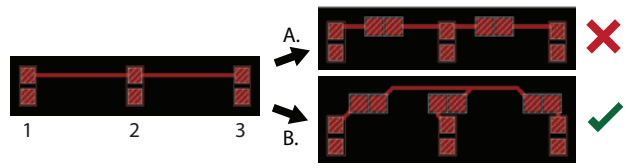

Figure 9: In many cases, adding jumper pads using only the designer's original routing cannot support the individual isolation of each component, as in (A), where elements 1 and 3 cannot be connected with 2 disconnected. Instead, Pinpoint must add new routes as in (B).

(2) **Remove problematic board traces.** Unlike in the schematic, Pinpoint must consider both the topology and the geometry of connections on the board layout to enable individual element isolation. While we aim to minimize deviation from the designer's original routing, many layouts require the removal or addition of traces, e.g. as shown in Figure 9, where the signal is routed "through" component 2. To preserve portions of the original routing while ensuring correctness, Pinpoint removes only wire segments incident upon elements, then routes with the remaining segments in place. Our limitations section describes in greater detail the considerations in preserving parts of the original routing.

(3) **Lay out new jumper pads.** To minimally impact the electrical properties of the circuit, placement of jumper pads should minimize the lengths of traces and number of vias required to route the resulting circuit. We define a cost function which weights free space on the board, distance from the original trace, and additional vias required in routing, then use a simulated annealing process to approximate the best available location for the new pad. Our current implementation uses a greedy approach for faster instrumentation, optimizing the location of one pad before considering the next. We leave it to future work to develop a global cost minimization procedure across all new pad locations to further optimize the placement.

(4) **Route new connections.** We finalize the instrumentation using EAGLE's default autorouter to connect the new pads into the circuit. The designer can adjust routing as desired once the instrumentation is complete.

Jig Fabrication

After instrumentation, Pinpoint adds #4-40 mounting holes in free space on the board design, then designs a jig board that routes pogo pin slots (above jumper pads on the DUT) to a header row (Figure 3.D) The designer assembles the jig by soldering in pogo pins, and connects it to the control board with a ribbon cable. As a lower-cost, rapid alternative, the designer can instead laser cut a jig template and inserts pogo

pins into friction-fit holes, connecting them to the control board with female jumper wires.

The Control Board

Pinpoint's custom testing hardware (Figure 10) serves to multiplex sites on the DUT for probing or signal injection, as well as to toggle connections across jumper pads. A dual-channel USB oscilloscope (Bitscope BS05 [5]) captures data from up to two signals on the DUT and transmits the traces via serial connection to the debugging interface. The BS05 also provides a waveform generator with interpolation of up to 1024 user-defined points. Pinpoint probes signals by multiplexing these channels (2x data capture, 1x signal injection) to any of up to 32 sites on an instrumented board (16 jumper pads, 2 ends per pad). We use three ADG732 32:1 analog multiplexers [9] (one for each channel), controlled by a Teensy 3.6 microcontroller [19] over USB serial connection.

The limits of Pinpoint's measurement capabilities are primarily set by specifications of the chosen oscilloscope. The BS05 has a 20 MHz analog bandwidth with a precision of 5 mV for signals below 1 MHz, and 20 mV at full bandwidth. It has a maximum sampling rate of 20 MSps on a timebase ranging from 1 μs/Div to 100 ms/Div. While it can be configured for a wide voltage range, our control board circuits are designed for 0-3.3V signals. These values lend themselves well to mixed signals in the 1 MHz range and below.

The control board also houses 16 optically-coupled solid-state relays (TLP241 [28]) that switch instrumented connections on the DUT. Each relay bridges the two sides of a jumper pad, such that closing the relay connects the two ends of the signal. Solid-state relays are faster, smaller, and more reliable than mechanical relays, at the cost of a more measurable impact on electrical properties of the circuit. However, optically-coupled relays offer better isolation between the circuit under test and control signals. The control board also breaks out the two sides of each relay to header pins beneath the ribbon cable. By connecting to these pins, designers can splice in external circuits on either side of a jumper pad.

The control board's integrated components cost approximately $25 total (100x quantity). The chosen microcontroller and USB oscilloscope cost $30 and $110 respectively.

Debugging Interface

Pinpoint's graphical interface (Figure 5) controls probing, signal injection, interruptible connections, and test logic via USB serial communication with the control board. Metadata and images imported from the instrumentation stage enable the designer to interact with a linked representation of their board and schematic.

Designers can create test signals either by configuring standard waveforms (sine, square, triangle, sawtooth), manually specifying interpolated points in a CSV, or recording

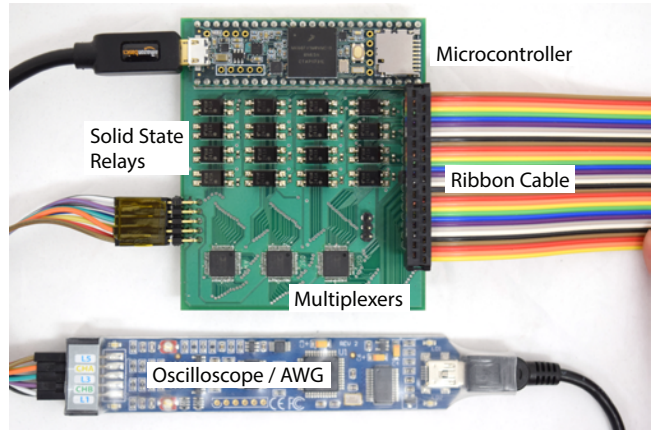

Figure 10: The control board is Pinpoint's custom testing hardware. Analog multiplexers switched by a microcontroller connect a USB oscilloscope and waveform generator to sites on the DUT to probe and inject signals. Solid state relays open and close the DUT's instrumented connections.

an observed signal from the board (e.g. to test the system's response to a problematic input).

Pinpoint currently supports tests of voltage, frequency, period, and continuity. Tests involve opening/closing connections on the board, injecting test signals, and then probing the result. For example, to check whether a resistor is well soldered on the DUT, Pinpoint opens the surrounding jumper pads to isolate the component, sets test voltages on one side of the component, and reads resulting voltages on the other side. The user imports tests from text files or authors them within the interface using a simple assertion syntax. For example, a unit test to detect a potential short circuit is written: `assert continuity <signal1> <signal2> false`, where "false" implies that the test passes if continuity is not found. Pinpoint automates continuity testing by generating a continuity check for each pair of signals on the board.

5 TECHNICAL EVALUATION

We evaluate our current implementation of Pinpoint by analyzing its potential impact on the electrical properties of the DUT and by exploring its potential to assist with a range of circuits and bugs.

Electrical Characteristics

Instrumentation of the DUT potentially alters its electrical characteristics through parasitic impedances, which may impact sensitive circuits. The total resistance across a jumper pad in its closed state – measured from the tip of one pogo pin, across the jig, wires, and relay, to the tip of the other pin – is 0.6Ω. The relay circuit itself (measured across pairs of pins on the control board) contributes 0.3Ω; the remainder of the resistance from the cables and jig can therefore be reduced

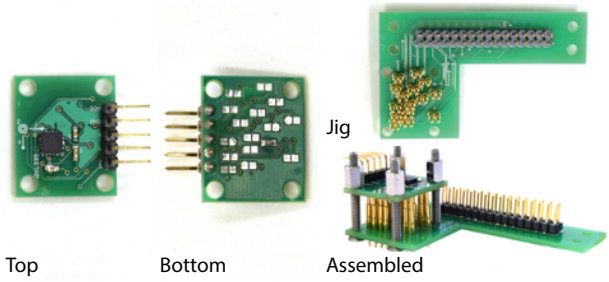

Figure 11: A fully instrumented analog 3-axis accelerometer PCB. Pinpoint can detect invisible short circuits beneath the small SMD package.

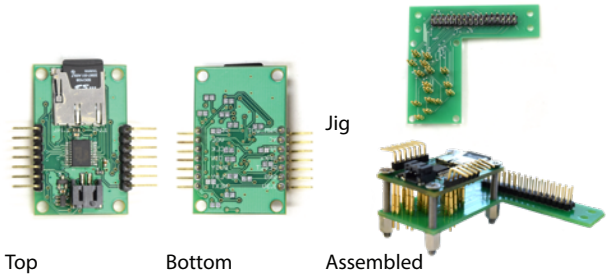

Figure 12: Debugging a digital MP3 player with instrumentation on I/O pins and power supply connections. Pinpoint can author functional tests that trigger playback with injected signals and verify that signals appear on the output.

by using larger wires and traces. The capacitance from pogo tip to pogo tip and across the relay circuit is 300pF, primarily in the off-state junction capacitance of the solid state relays. Finally, as with any oscilloscope, probing itself impacts the DUT. The USB oscilloscope in our implementation has a per-channel impedance of 1MΩ / 10pF.

Using these values, designers can calculate the anticipated impact of Pinpoint on sensitive signals. Potential risks of added resistance include: amplification of voltage changes due to transients and leakage current, changes in rise/fall times, disruption of impedance matching, etc. The parasitic capacitance is most likely to impact high frequency (e.g. RF) circuits, feedback-based amplification circuits, etc. Finally, the added traces and connectors could provide vectors for coupling/crosstalk on sensitive signals. Throughout our usage of Pinpoint on both analog and digital signals, we encountered only one breakdown caused by our instrumentation load, when testing Pinpoint's limits on a 16 MHz crystal oscillator sensitive to small changes in capacitance.

Demonstrative Applications

We instrumented and fabricated manufacturer-provided board designs for several representative, commercially available PCBs from SparkFun to explore Pinpoint's utility across different types of bugs and boards. Table 1 lists relevant parameters of each board, including a description of components instrumented.

Demonstration 1: Accelerometer. We first instrumented all signals on a single analog IC PCB – the ADXL335 3-axis accelerometer [24]. The value of a capacitor on each channel sets its bandwidth (Figure 11).

Using Pinpoint, the designer can quickly compare and record outputs from each axis to characterize any errant behaviors. Automatic issue detection can detect solder bridges hidden beneath the small IC package. Splicing in a breadboard enables rapid exploration of different capacitor values to calibrate the bandwidth of each channel.

Demonstration 2: MP3 Player. We next selected a digital circuit to demonstrate how Pinpoint can inject signals to send control signals and debug digital logic. The circuit uses the WTV020SD audio playback IC and supports simple controls such as play/pause/next [20]. We instrumented these control signals, the device's output pins, and the power and ground outputs of the on-board voltage regulator (Figure 12).

Debugging often requires localizing a fault across a cascade of hardware and software stages. If our device fails to play audio when triggered, we can author tests to determine whether the issue lies in software (we inject control signals directly onto the board, bypassing software), in the routing of input signals to the control pins of the IC (we inject control signals directly to the pins of the IC), in the IC itself (we verify that the "busy" line of the IC goes high as expected when playback is triggered), or in the routing of outputs to a speaker (we verify that expected audio signals are seen on the output pins). Having instrumented the power regulator on the board, we can quickly swap it out for an external source to debug power issues.

Demonstration 3: Digital Alarm Clock. Next, we instrumented a circuit with physical controls to explore the use of Pinpoint in debugging interaction events. We chose a digital alarm clock with an on-board microcontroller and various buttons and switches to configure times and alarms [21]. We instrumented the input elements, as well as the individual control lines for a segment on the digital display (Figure 13).

Physical switches and buttons can give rise to unique bugs. For example, many input errors result from "bouncing" – when the switching of a physical element creates multiple brief edges during the transition. We can test our software free from bouncing by bypassing the physical switches to inject clean test signals into the microcontroller. Then, we can characterize the bouncing behavior by performing a oneshot capture to record a button press event. After updating our code to filter the bouncing correctly, we can create a unit test that injects an artificially "bounced" signal.

Table 1: Demonstrative Application Board Parameters

Application	# of Instrumented Sites / # of Possible Sites	Signal Types	Instrumented Components	Representative Debugging Interactions
Accelerometer	16 / 16	Analog	Sensor, power rails, bandwidth capacitors	Automated continuity checks, splicing in prototyped circuits
MP3 Player	16 / 41	Mixed	Control pins, outputs, voltage regulator	Injecting control signals to bypass software, replacing power sources
Alarm Clock	15 / 42	Digital	Physical switches, LED segment control lines	Transient analysis (oneshot capture), signal record and replay
FM Radio	15 / 74	Mixed	Subcircuit boundaries, power rails, board I/O	Subcircuit integration testing, replacing subcircuits with known working elements, signal record and replay

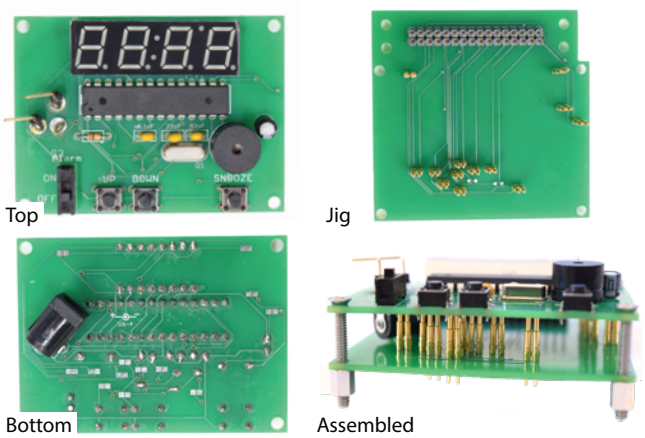

Figure 13: A clock with instrumentation on physical input elements. Pinpoint can help analyze switch bouncing by pausing, recording, and replaying button presses.

Demonstration 4: FM Radio. Finally we demonstrate how Pinpoint facilitates integration testing by isolating functional subcircuits on a board. We selected an FM radio tuner with on-board audio amplifier [22]. Our instrumentation isolated each subcircuit – the FM receiver, audio amplifier, headphone jack, and header pins providing I/O to the board – including the power, ground, and I/O lines of each (Figure 14).

By isolating each subcircuit, we can debug each in isolation, or test a problematic stage with a working replacement. For example, we can replace the FM receiver with an external audio source to test the audio amplifier, or we can verify the FM receiver by connecting its outputs to an external speaker. We can store a recorded a signal from the FM receiver to use as a unit test of the audio amplifier.

6 INFORMAL FEEDBACK

To solicit formative critiques of our system from active practitioners, we engaged (6) students and researchers (4 male, ages 21-29) with circuit and PCB development experience

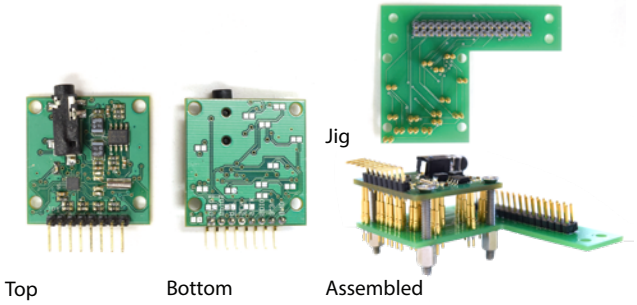

Figure 14: Debugging an FM Radio tuner with interruptible connections between each subcircuit. Pinpoint can disconnect the FM receiver from the circuit and replace it with a known good audio source to test the audio amplifier subcircuit in isolation.

in a guided exploration of Pinpoint. Two of these users had provided unstructured interviews in the early needfinding stages of our work. We led each user through a simple debugging task (the game controller, as described in our example scenario) to demonstrate Pinpoint's primary features, then gave them a new faulty board (our analog accelerometer PCB with a bandwidth capacitor shorted to ground) to freely debug in a thinkaloud fashion. To encourage creative problem solving, we instructed users not to find the issue using automated continuity checking.

All users isolated the target bug through a variety of strategies. Some opened connections in an exploratory fashion to localize which elements resolved the short when removed from the circuit. Others authored continuity checks for potentially shorted sites. One user spliced cables across different parts of the circuit to identify which connections introduced the short into a probed region.

Our users offered appreciative remarks for each of Pinpoint's features, showing particular interest in interactive probing, subcircuit isolation, test signal injection, and splicing of new circuits. Several users attested that Pinpoint would

greatly accelerate "the hardest part" of their debugging – localizing the error to a single site – over their current methods (manual probing with an oscilloscope). Some envisioned uses of the tool unplanned by the authors, such as leveraging Pinpoint's hardware interface as a way to quickly break out signals on their board for use with a logic analyzer.

The users also mentioned several desired additions, such as swept-frequency tests or the ability to designate pins as having either digital or analog behavior. Many users were particularly interested in the embedding of expert knowledge as context-aware debugging assistance. For example, some mentioned that while Pinpoint *empowered* them with new testing capabilities, they also expected it to *educate* them on which tests might prove useful in debugging their issue.

7 LIMITATIONS AND FUTURE WORK

Beyond optimizations to Pinpoint's hardware elements (e.g. using relays optimized for low junction capacitance to reduce parasitic impedance), Pinpoint's most fundamental challenge lies in preserving the original routing of the board design during instrumentation. Because autorouted designs are typically suboptimal, designers often route their boards manually, and could be disincentivized from using Pinpoint if it significantly alters their carefully laid plans or limits their understanding of the board. Improvements to our layout algorithm can increase the total fraction of the original routing that is preserved. However, in many cases, such as in Figure 9, the original routing does not contain paths necessary to support individual isolation of each component from the circuit. Rather than limit Pinpoint to circuits routed in particular configurations, we allow it to perform rewirings of the board that the designer can subsequently adjust.

Scaling Considerations for Large/Complex Boards

While we expect users to exercise discretion on which signals to instrument in large designs, we can analyze the scalability of our approach for more complex designs than those fabricated as above. We characterize the requirements for *fully instrumenting* a board in terms of: 1) number of sites and 2) space requirements.

While we chose 16 sites of instrumentation in our prototype implementation, we can extend our control board by adding relays/multiplexers (scaling linearly with the number of sites) to support more complex designs. We ran our instrumentation algorithm over representative circuit sets to gather data on the typical number of sites required to instrument different types of PCBs. These included the SparkFun Sensor Kit [23] ($\tilde{x}$ = 30, min = 13, max = 66), Arduino Starter Kit [3] ($\tilde{x}$ = 14, min = 3, max = 25), and Synthrotek Audio Synthesis boards [26] ($\tilde{x}$ = 64.5, min = 15, max = 436).

Pinpoint requires space to route added traces and space on the bottom layer of the board for pads. As additional board layers can facilitate routing, pad area is typically the limiting factor. The sum of the areas of each pad (3.9 mm^2, plus vias if necessary) approximates the required free space on the bottom layer. If needed, the designer can widen the board edges for debugging, then fabricate with the original dimensions for a padless production version. While we currently also add mounting holes (7.3 mm^2 each) in the corners of boards for mating the DUT and jig, other mating methods (such as 3D-printed enclosures) can obviate this requirement.

Finally, we consider the implications on scaling of any steps involving manual labor, such as assembling the jig. In addition to the solder-free jig fabrication method described in our Implementation section, we are exploring new methods for creating jigs without assembled elements. For example, we can fabricate the jig as a flexible PCB, which attaches to and interfaces with pads on the DUT via a single sheet of Z-axis conductive transfer tape. Other techniques can also reduce manual soldering across jumper pads when debugging has finished (e.g. using a reflow oven or a conductive ink pen).

New Opportunities for Pinpoint

Future iterations of Pinpoint can benefit from new debugging modes, such as logic analysis, as well as intelligent design assistance features. For example, using metadata from the user's board designs, Pinpoint could automatically detect and configure for digital, analog, or mixed signal analysis in a context-aware fashion. Pinpoint could also suggest groupings of subcircuits to assist the user in allocating instrumentation sites in large designs.

Moving forward, we see potential for Pinpoint not just as a tool but as a platform, to be used in education or research. With the addition of network capabilities, instructors and collaborators can use Pinpoint as a remote debugging lab, authoring tests to quickly diagnose errors in an online help session. In our own lab, we added logging to Pinpoint and have begun using it to study questions of debugging strategies and cognitive models.

8 CONCLUSION

Pinpoint facilitates PCB debugging by addressing core issues in access, isolation, and iteration. By automatically instrumenting board designs and simplifying the fabrication of a hardware-software interface, it provides designers with new in-circuit functional testing methods. With continued growth in the space of personal electronic fabrication, we see Pinpoint as an important, yet early, step towards accessible computational support of PCB debugging tasks.

9 ACKNOWLEDGEMENTS

We thank Mathieu LeGoc, Alexa Siu, and Jingyi Li for their guidance. This work was supported by an NDSEG fellowship.

REFERENCES

[1] Fraser Anderson, Tovi Grossman, and George Fitzmaurice. 2017. Trigger-Action-Circuits. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology - UIST '17*. ACM Press, New York, New York, USA, 331–342. https://doi.org/10.1145/3126594.3126637

[2] apertus. 2018. DIY Flying Probe / Automated Optical Inspection. https://www.apertus.org/diy-flying-probe-AOI

[3] Arduino. 2018. Arduino Starter Kit Multi-Language. https://store.arduino.cc/usa/arduino-starter-kit

[4] Autodesk. 2018. EAGLE PCB Design Software. https://www.autodesk.com/products/eagle/overview

[5] BitScope Designs. 2018. BitScope Micro Model 5. http://www.bitscope.com/product/BS05/

[6] Keith Brindley. 1991. *Automatic Test Equipment*. Elsevier Science. 241 pages.

[7] Varun Perumal C and Daniel Wigdor. 2015. Printem. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology - UIST '15*. ACM Press, New York, New York, USA, 243–251. https://doi.org/10.1145/2807442.2807511

[8] Ying Chen and Can Wu. 2017. The hot spot transformation in the research evolution of maker. *Scientometrics* 113, 3 (12 2017), 1307–1324. https://doi.org/10.1007/s11192-017-2542-4

[9] Analog Devices. 2018. ADG732 Datasheet and Product Info. http://www.analog.com/en/products/adg732.html

[10] Digilent. 2018. Electronics Explorer: All-in-one USB Oscilloscope, Multimeter & Workstation. https://store.digilentinc.com/electronics-explorer-all-in-one-usb-oscilloscope-multimeter-workstation/

[11] Daniel Drew, Julie L. Newcomb, William McGrath, Filip Maksimovic, David Mellis, and Björn Hartmann. 2016. The Toastboard. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology - UIST '16*. ACM Press, New York, New York, USA, 677–686. https://doi.org/10.1145/2984511.2984566

[12] Dimitris Gizopoulos. 2006. *Advances in electronic testing: challenges and methodologies*. Vol. 27. Springer.

[13] Pragun Goyal, Harshit Agrawal, Joseph A. Paradiso, and Pattie Maes. 2013. BoardLab. In *Proceedings of the Adjunct Publication of the 26th Annual ACM Symposium on User Interface Software and Technology - UIST '13 Adjunct*. ACM Press, New York, New York, USA, 19–20. https://doi.org/10.1145/2508468.2514936

[14] Björn Hartmann, Scott R. Klemmer, Michael Bernstein, Leith Abdulla, Brandon Burr, Avi Robinson-Mosher, and Jennifer Gee. 2006. Reflective physical prototyping through integrated design, test, and analysis. In *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology - UIST '06*. ACM Press, New York, New York, USA, 299. https://doi.org/10.1145/1166253.1166300

[15] JTAG Technologies. 2018. Boundary Scan In-Circuit Programming. https://www.jtag.com/

[16] Will McGrath, Daniel Drew, Jeremy Warner, Majeed Kazemitabaar, Mitchell Karchemsky, David Mellis, and Björn Hartmann. 2017. Bifröst: Visualizing and Checking Behavior of Embedded Systems Across Hardware and Software. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology - UIST '17*. ACM Press, New York, NY, USA, 299–310. https://doi.org/10.1145/3126594.3126658

[17] 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 - DIS '16*. ACM Press, New York, New York, USA, 1270–1281. https://doi.org/10.1145/2901790.2901833

[18] National Instruments. 2017. LabVIEW. http://www.ni.com/labview/

[19] PJRC. 2017. Teensy USB Development Board. https://www.pjrc.com/teensy/index.html

[20] SparkFun Electronics. 2018. SparkFun Audio-Sound Breakout - WTV020SD. https://www.sparkfun.com/products/11125

[21] SparkFun Electronics. 2018. SparkFun ClockIt. https://www.sparkfun.com/products/10930

[22] SparkFun Electronics. 2018. SparkFun FM Tuner Evaluation Board - Si4703. https://www.sparkfun.com/products/12938

[23] SparkFun Electronics. 2018. SparkFun Sensor Kit. https://www.sparkfun.com/products/13754

[24] SparkFun Electronics. 2018. SparkFun Triple Axis Accelerometer Breakout - ADXL335. https://www.sparkfun.com/products/9269

[25] Evan Strasnick, Maneesh Agrawala, and Sean Follmer. 2017. Scanalog: Interactive Design and Debugging of Analog Circuits with Programmable Hardware. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology - UIST '17*. ACM Press, New York, New York, USA, 321–330. https://doi.org/10.1145/3126594.3126618

[26] Synthrotek. 2018. Synthrotek. http://store.synthrotek.com/

[27] M. Tawfik, E. Sancristobal, S. Martin, R. Gil, G. Diaz, A. Colmenar, J. Peire, M. Castro, K. Nilsson, J. Zackrisson, L. Hkansson, and I. Gustavsson. 2013. Virtual Instrument Systems in Reality (VISIR) for Remote Wiring and Measurement of Electronic Circuits on Breadboard. *IEEE Transactions on Learning Technologies* 6, 1 (1 2013), 60–72. https://doi.org/10.1109/TLT.2012.20

[28] Toshiba. 2018. Photorelays. https://toshiba.semicon-storage.com/eu/product/opto/photorelay.html

[29] 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. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology - UIST '16*. ACM Press, New York, New York, USA, 687–695. https://doi.org/10.1145/2984511.2984527

[30] Te-Yen Wu, Mike Y. Chen, Bryan Wang, Jiun-Yu Lee, Hao-Ping Shen, Yu-Chian Wu, Yu-An Chen, Pin-Sung Ku, Ming-Wei Hsu, and Yu-Chih Lin. 2017. CircuitSense. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology - UIST '17*. ACM Press, New York, New York, USA, 311–319. https://doi.org/10.1145/3126594.3126634

[31] Te-Yen Wu, Hao-Ping Shen, Yu-Chian Wu, Yu-An Chen, Pin-Sung Ku, Ming-Wei Hsu, Jun-You Liu, Yu-Chih Lin, and Mike Y. Chen. 2017. CurrentViz. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology - UIST '17*. ACM Press, New York, New York, USA, 343–349. https://doi.org/10.1145/3126594.3126646