Scanalog: Interactive Design and Debugging of Analog Circuits with Programmable Hardware

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

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

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

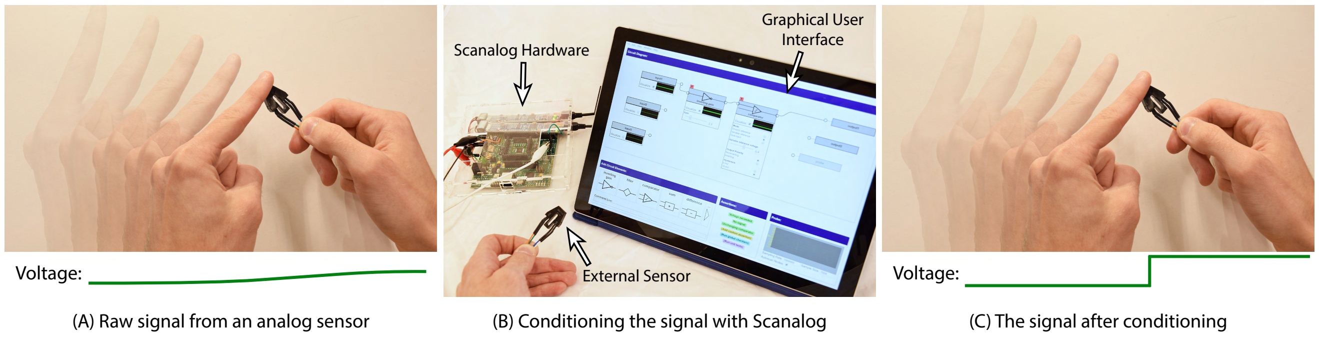

Voltage:
Scanalog Hardware
Graphical User Interface
External Sensor
Voltage:
(A) Raw signal from an analog sensor
(B) Conditioning the signal with Scanalog
(C) The signal after conditioning

Figure 1. Scanalog integrates reconfigurable analog hardware, programmable data capture and replay, and a high-level software interface to support the design and debugging phases of analog circuit work. Here, raw signals from an IR photoelectric sensor are conditioned using gain and comparator modules to produce a clean digital output, enabling proximity sensing in interactive devices.

ABSTRACT

Analog circuit design is a complex, error-prone task in which the processes of gathering observations, formulating reasonable hypotheses, and manually adjusting the circuit raise significant barriers to an iterative workflow. We present Scanalog, a tool built on programmable analog hardware that enables users to rapidly explore different circuit designs using direct manipulation, and receive immediate feedback on the resulting behaviors without manual assembly, calculation, or probing. Users can interactively tune modular signal transformations on hardware with real inputs, while observing real-time changes at all points in the circuit. They can create custom unit tests and assertions to detect potential issues. We describe three interactive applications demonstrating the expressive potential of Scanalog. In an informal evaluation, users successfully conditioned analog sensors and described Scanalog as both enjoyable and easy to use.

ACM Classification Keywords

H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous

UIST 2017, October 22–25, 2017, Quebec City, QC, Canada
© 2017 ACM. ISBN 978-1-4503-4981-9/17/10…$15.00
DOI: https://doi.org/10.1145/3126594.3126618

Author Keywords

Scanalog; Scanalogue; design tools; debugging; analog circuits; electronics; prototyping; FPAA; Field Programmable Analog Array; programmable analog hardware

INTRODUCTION

Analog electronics are widely used to sense and act on signals from the real world. For example, a light-seeking robot might employ analog circuits to *condition* signals from an photocell to sense brightness, *interface* those signals with digital logic using an analog-to-digital converter, and then *drive* outputs such as motors. Analog and mixed-signal stages occur in most end-user devices, and are particularly advantageous in interactive devices to process continuous signals with speed, precision, and low power.

However, designing analog circuits is difficult for a number of reasons. Among these are:

1. ***Physical construction of circuits is slow and error-prone.*** Wiring mistakes and poor connections introduce hard-to-detect bugs even in correct designs. [7].

2. ***Real circuits often do not match simulations***, e.g. due to noise, faulty hardware, and limitations of real components.

3. ***Changes to an analog system often propagate to affect other components***, making it difficult to isolate bugs or detect the failures of previously functional subunits. Without the ability to automatically monitor parts of the circuit with assertions or unit tests, the user must take backtrack repeatedly to find sources of error.

4. ***The inability to readily observe the internal state of the circuit*** results in a slow, complex, and error-prone debugging process in which the designer manually probes with an oscilloscope to gather observations.

These issues hinder the ability of designers to iteratively design and debug circuits, as each step in the iterative cycle (observing the behavior, diagnosing bugs, redesigning the circuit, and implementing the new design) is slow and laborious [19].

One technology that can address some of these difficulties is the Field-Programmable Analog Array (FPAA). The FPAA is a *programmable* analog hardware platform. Much like an FPGA (Field Programmable Gate Array) enables the programming of digital logic, FPAAs utilize programmable resources to connect discrete analog components into a variety of circuits. An FPAA is typically accompanied by a software design tool, which lets users specify a circuit design, flash the design onto the board, then test the board in their physical application, iterating as necessary. FPAAs address two of the aforementioned difficulties of analog circuit design: (1) Users can rapidly make modifications to a circuit without the need to mechanically assembly components, accelerating the iterative process and reducing mis-wiring errors. (2) Users can design the circuit in tandem with real inputs, eliminating errors in moving from simulation to application.

Despite these advantages, FPAAs remain unhelpful in assisting the user to better observe and debug the behavior of their circuit. To support an iterative workflow, we desire not only that the user can readily make changes using *direct manipulation*, but also that they can readily observe the effects of those changes with *immediate feedback*. We present Scanalog ("Scan" + "Analogue", Figure 1), an interactive, high-level design tool which interfaces with an FPAA to support the processes of designing, building with, and understanding analog electronics. Scanalog combines rapidly programmable hardware, integrated sensing of internal signals, and arbitrary waveform generation to address the difficulties of analog design in the following ways:

- Provides the user with direct manipulation of the hardware in real-time while observing its response to real input. This tightened feedback loop allows for rapid iteration and enables the user to "think by doing."
- Facilitates understanding of internal state by displaying signals from all points in the working circuit in real-time. These visualizations provide immediate feedback of the behavior of the entire circuit as the user makes changes.
- Enables the use of built-in and user-defined assertions to automatically detect issues during design. This eliminates the need to manually search for the source of an error when new modifications to the circuit impact previously functional elements.
- Enables the user to record real inputs and subsequently replay them through the hardware to perform unit testing. This aids the user in verifying the correctness of their hardware without simulation or the need to repeatedly and manually provide inputs.

RELATED WORK

Circuit Design Tools

Scanalog draws from a number of tools used by experts in the design and prototyping of circuits. Graphical circuit design tools such as EAGLE [5], Altium Designer [2], or Fritzing [17] assist the user in adding components to and wiring a circuit schematic. Users can then lay out designs for fabrication to create functional hardware. Circuit design tools typically also include simulators, such as SPICE [20], which help the user to predict the response of their system to a variety of conditions. Like Scanalog, many of these simulators and graphical circuit design tools enable interactive adjustment of circuit parameters. However, they do not operate on live signals, and do not configure hardware that can be tested in the context of the application.

An interface designed by Victor [23] also visualizes simulated signals in real time throughout an interactive virtual circuit, inspiring the design of our system. Scanalog implements these interactions on real hardware and signals.

LabVIEW [16] is a data acquisition tool that can automate the processes of observing input signals and processing them using a dataflow programming interface. In addition, it can be used to program the behavior of Digital Signal Processors (DSPs). Like FPAAs, DSPs can be programmed to interface with and manipulate continuous signals in various ways. However, DSPs rely on quantizing an analog signal and manipulating it using digital logic, yielding limitations in terms of resolution, frequency range, power, etc. as compared to analog components. While LabVIEW and other data acquisition tools can display real-time signals from the circuit similarly to Scanalog, they must be manually configured with a specific probe for each site.

Finally, design tools for FPAAs such as AnadigmDesigner [4] and Cypress PSoC Creator [10] interface directly with programmable analog hardware. Their graphical user interfaces enable the user to lay out a design consisting of high-level modules, connect them together, and set their parameters. When finished with the design, the user can then flash the circuit onto the FPAA. These circuits are typically static once flashed, with the exception of a small set of exposed parameters (e.g. gain) which the user can control by developing a custom application alongside their circuit. For example, using AnadigmDesigner, a user can design a circuit topology, set values for each configurable parameter, flash the FPAA, and probe the output to verify their design. Then, when the circuit requires an adjustment, the user must return to the software design, set new values, reprogram, and probe once more. Scanalog instead provides direct manipulation of the FPAA by immediately reflecting changes within the interface onto the hardware, and visualizes within the interface the real-time signals passing through the hardware.

Debugging and Prototyping in Physical Computing Tasks

Prior work has explored a number of techniques for more interactive development and debugging in electronic device design. The Toastboard [12] and the Visible Breadboard [21] provide live debugging assistance while prototyping circuits

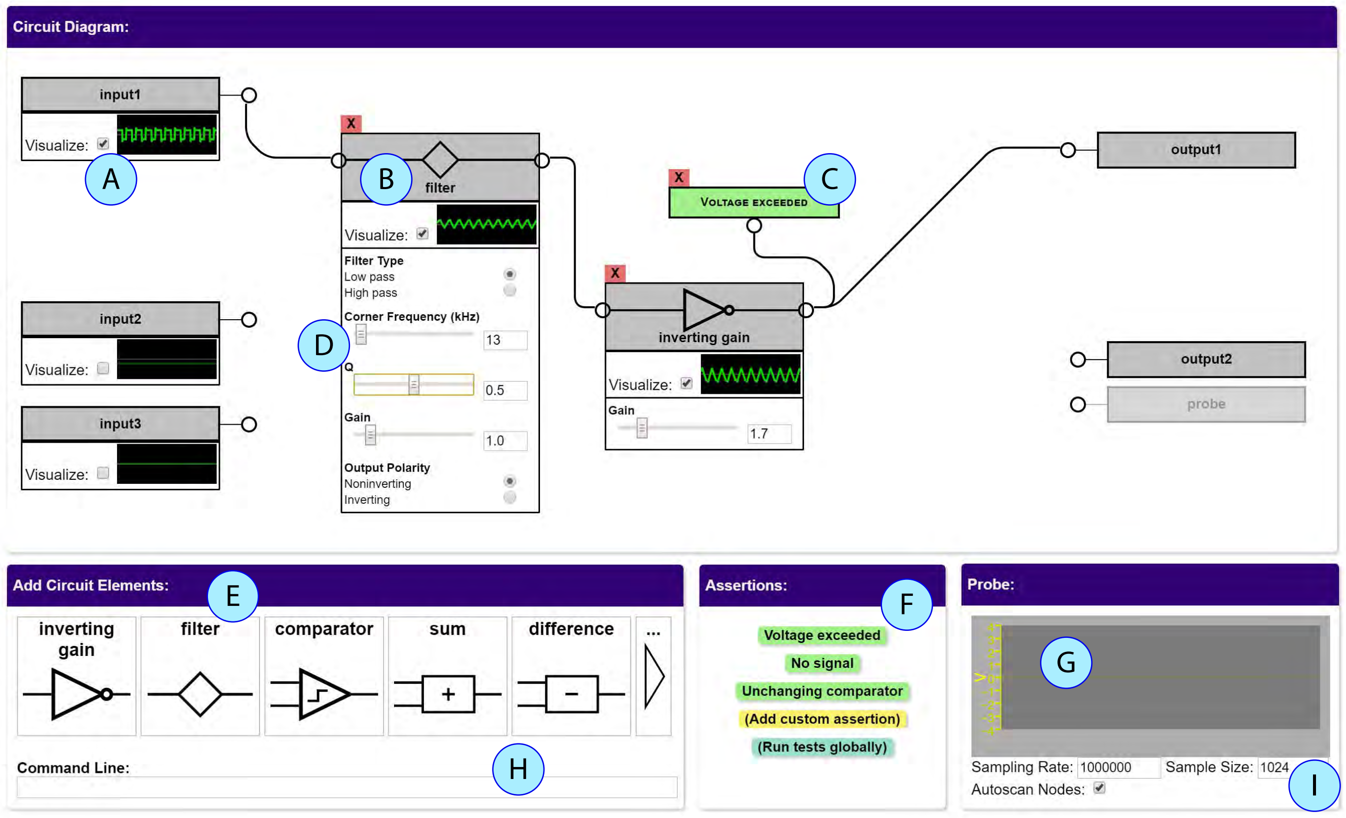


Figure 2. Scanalog's graphical user interface. A: Live visualizations for each module. B: Module instantiated in the workspace. C: Assertion instantiated in the workspace. D: Tunable parameters of a module. E: Add Circuit Elements palette. F: Assertion/test palette. G: Visualization for manual probe mode. H: Optional command line. I: Oscilloscope settings.

on a breadboard. However, because these systems do not leverage programmable hardware, they still require users to physically construct circuits at a manual pace. Moreoever, as tools designed to debug digital applications, they are largely unable to interpret continuous signals, making them unsuitable for use in analog circuit design.

Researchers have also developed systems that visualize the internal behavior of a circuit. For example, LightUp, Projectron Mapping, and Flow of Electrons use augmented reality to display information about a circuit in-situ ([1], [8], [9]). Gamified systems like TagTiles aim to help users gain a familiarity with electronics and their principles through play [22]. While pedagogically valuable, these interfaces visualize a limited set of behaviors and do not serve as general-purpose analog circuit design tools.

Other tools have focused on helping users to more efficiently prototype circuits. CircuitStack [25] provides breadboard-like prototyping on top of software-designed PCB layers, combining the error-resilience of printed circuits with the flexibility of a breadboard. Similarly Exemplar [13] and PICL [15] leverage "Programming by Demonstration" to rapidly configure microprocessors to give desired outputs in response to sensor data. Finally, toolkits like Phidgets [14], Calder [18], littleBits [6], and .NET Gadgeteer [24] consist of modular components intended to jump-start the creation of interactive devices. These tools can reduce the time needed to design and prototype functional hardware, but lack customizability beyond their pre-defined functionality.

WALKTHROUGH OF THE SCANALOG SYSTEM

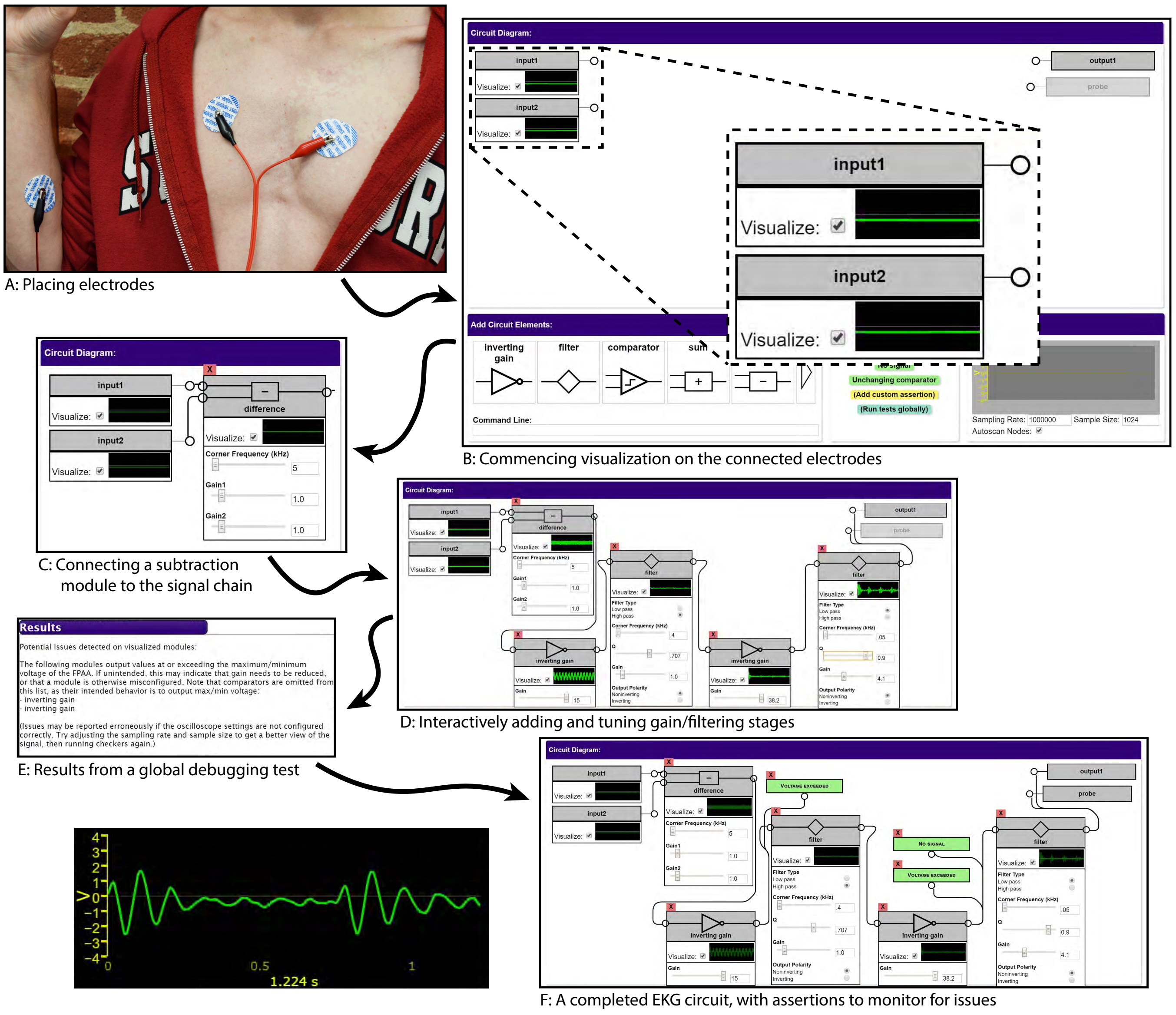
To illustrate the features of Scanalog, consider the following example task taken from an introductory lab assignment given to University students (Figure 3):

A student, Sam, is tasked with building an EKG, a device which can visualize electrical activity in the heart. He learns that EKGs work by picking up and amplifying the difference in electrical potential across the body. Knowing this, he connects two electrodes to inputs of the FPAA board, attaching the other ends to his chest (Figure 3.A).

Users begin designing with Scanalog by connecting its hardware to their PC and power source, and opening the interface. They can then connect to the FPAA board the input and output connections for their application. All changes to the hardware are thereafter made with Scanalog's dataflow software interface (Figure 2) or a built-in command line (Figure 2.H).

Real-Time Visualization of All Circuit Nodes

As Sam connects the electrodes, a visualization of the signal immediately appears on the screen next to the input modules (Figure 3.B).



Results

Potential issues detected on visualized modules:

The following modules output values at or exceeding the maximum/minimum voltage of the FPAA. If unintended, this may indicate that gain needs to be reduced, or that a module is otherwise misconfigured. Note that comparators are omitted from this list, as their intended behavior is to output max/min voltage:
- inverting gain
- inverting gain

(Issues may be reported erroneously if the oscilloscope settings are not configured correctly. Try adjusting the sampling rate and sample size to get a better view of the signal, then running checkers again.)

Figure 3. An example workflow demonstrating the features of Scanalog in the development of an EKG.

Scanalog's principal mode of assistance is the real-time visualization of live signals from all modules in the circuit. In our example, Scanalog automatically probes signals from the electrodes from the moment they are connected, rendering them as waveforms alongside the corresponding input in the graphical user interface (Figure 2.A). The user can enable or disable live visualization on the output of each modular stage within the circuit. Users can also switch from *autoscan mode* into *manual probe mode*, allowing them to capture the output of any particular module in the circuit with a higher resolution and refresh rate. As the user advances their design and tunes parameters, they can watch in real-time how the behavior of the entire circuit changes.

Immediately Programmed High-Level Modules

Next, Sam knows that the EKG must take the difference between the two signals, so he selects the Subtraction module from the Add Circuit Elements palette (Figure 2.E) and adds it to the circuit. Scanalog immediately reconfigures the FPAA with resources that subtract the two inputs (Figure 3.C).

Scanalog provides users with high-level modules encapsulating common subcircuits. Available modules were selected by observing the components used in analog circuits constructed in introductory and intermediate engineering classes at Stanford University, and include gain, summation, subtraction, high-pass filtering, low-pass filtering, and comparator operations. Changes to the topology or parameters of the working circuit are immediately reflected in Scanalog's underlying hardware, so that users can try various options and observe their effects on the signal in real time.

Tuning Module/Subcircuit Parameters

Though the electrodes are connected, the result of the subtraction shows no visible signal. Sam hypothesizes that the amplitude of the signal is too low to see, so he adds a Gain module to the design. Unsure of the exact amount of amplification, he scrubs the gain parameter of the module while observing the output. As the gain increases, he observes a signal. However, the signal is mostly noise, and so he correspondingly modifies the design by adding and tuning a low-pass and a high-pass filter. After filtering, the signal once more appears to require amplification, so Sam continues adding and tuning gain stages in an iterative fashion as the EKG signal appears increasingly well conditioned (Figure 3.D)

For each module, Scanalog provides interactive controls for their configurable parameters (Figure 2.D). Because changes are reflected instantly on the hardware and in the live visualizations, users can simply scrub parameters until the ideal behavior is achieved, rather than repeatedly guessing or calculating values. This direct parameter manipulation helps the user to build a holistic understanding of the circuit and facilitates iteration.

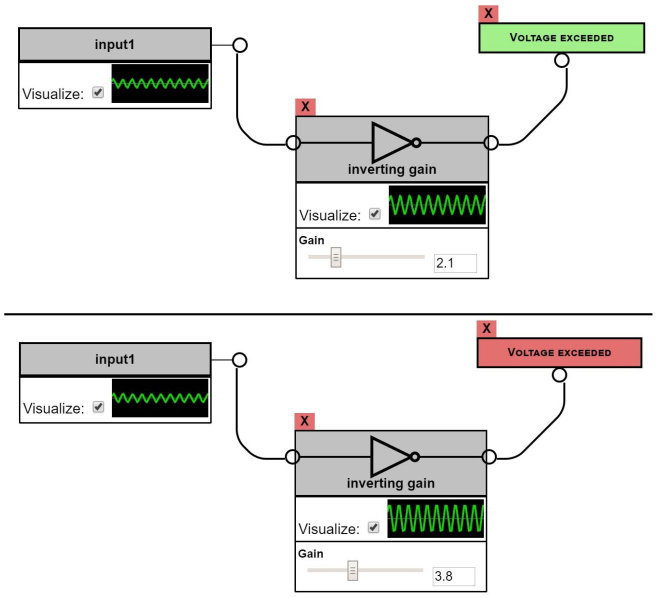


Figure 4. Adding in assertions to monitor invariants while designing. Here, the system detects if the monitored output exceeds the operating voltage of the hardware, resulting in clipping of the signal.

Debugging With Customizable Tests

As Sam interactively tunes parameters, he notices that certain parameter combinations result in the signal appearing distorted or "clipped." Unsure of the issue, he runs a pass of built-in global debugging tests over the circuit. Scanalog reports that the problematic module is exceeding the operating voltage of the board, and that gain should be reduced at that stage. (Figure 3.E).

Scanalog stores data captured for visualization to additionally provide automated debugging features. Scanalog is capable of scanning the current circuit for likely causes of issues, as defined by a customizable list of *global debugging tests* (Figure 2.F). Users can write their own tests in JavaScript using a built-in interface within the application.

Signal Assertions

Though he resolves the voltage clipping issue, Sam is worried that continuing to tune the parameters of the circuit will cause the same problem elsewhere. To alert himself in this case, he adds assertions to his workspace, hooking them up to the output of the later gain stages. When later tuning causes the output of this module to reach the operating voltage of the board, the test visually alerts him to the issue.

Scanalog provides the ability to add software-like assertions that check when the signal violates properties that should remain invariant (Figure 4). The checks are performed in real time, and assertions update visually to indicate a violation. As changes to an analog component often affect behaviors in other parts of the circuit, assertions are particularly useful in detecting and preventing regressions introduced by modifications to the circuit. Scanalog contains a built-in set of assertions for basic issues (e.g. no signal on the output), and the user can easily define custom assertions using an editor within the application.

Recording, Replaying, and Unit Testing Signals

As Sam seeks to finalize the design, he repeatedly compares several combinations of parameters to determine which allows him to tune the final filtering stage most effectively. Rather than repeatedly switching back and forth the settings on the many modules earlier in the design, Sam records for each combination of parameters the output of the gain stage just before the final filter. Then, he uses Scanalog's playback functionality to generate each signal at the input of the board on demand. This allows Sam to find the optimal set of parameters and finish the design.

Scanalog allows users to record signals from the input or any module within the circuit, and subsequently play them back into the input of the FPAA on command. This feature serves two main functions: As a debugging tool, this allows the user to reliably produce a problematic signal, much like they can manually simulate a corner case while debugging software. Moreover, it enables the use of unit tests. For example, a user designing an interactive system that classifies various gestures can record an example of each gesture, and quickly verify that the system recognizes all gestures correctly after each modification to the design. When run, unit tests iterate through the collection of recorded signals, verify each assertion for each signal, and report any test failure.

At the end of this example, the FPAA is configured as a working EKG, and the student can directly embed it in his application. Alternatively, Scanalog can translate the circuit into a standard representation with discrete components and export it to a PCB design tool (such as Fritzing) to facilitate fabri-

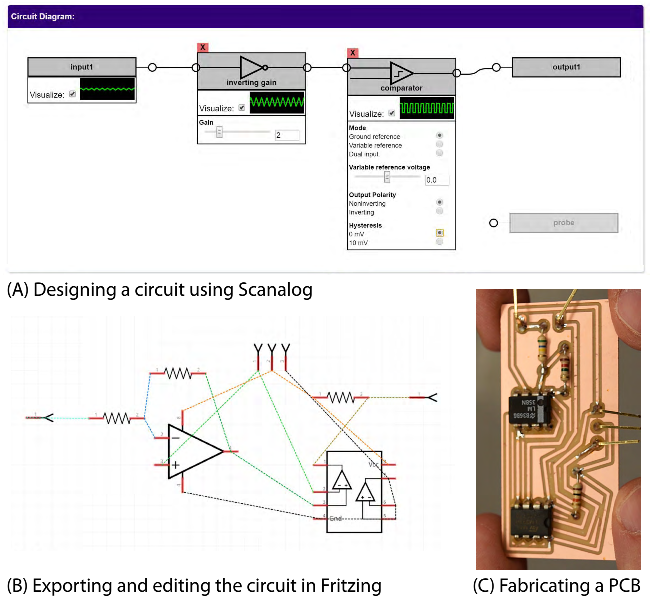


Figure 5. Exporting a circuit from Scanalog (top) into Fritzing (bottom left), and then fabricating the designed circuit with standard components on a milled pcb (bottom right).

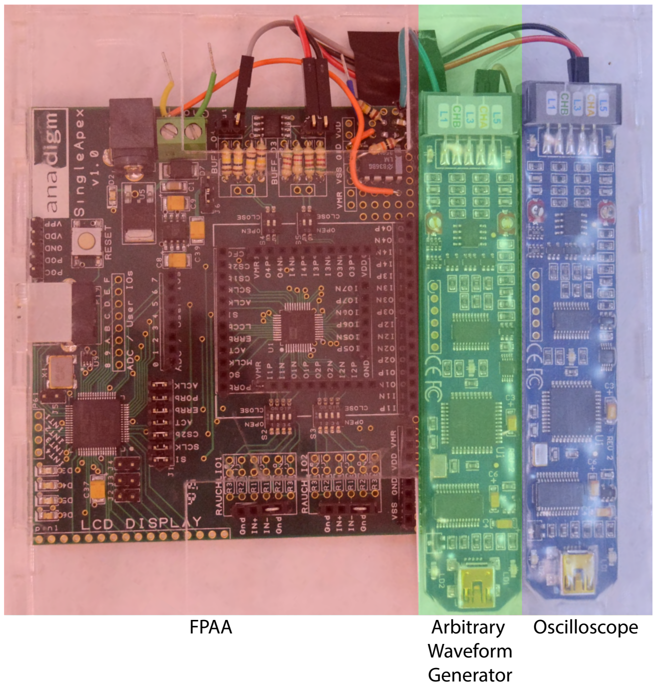


Figure 6. Scanalog's hardware components.

cation (Figure 5). The output of the EKG from this example application is visualized in Figure 3.F.

IMPLEMENTATION

In addition to its graphical user interface, Scanalog has three primary components: an FPAA, a programmable USB oscilloscope, and an arbitrary waveform generator (Figure 6). Figure 7 shows the communications between these components in a functional diagram.

Dynamic Configuration of the FPAA

The FPAA is the core hardware that serves as the user's analog circuit. We use the Anadigm AN231E04 [3] due to its ability to flash new configurations very rapidly. The AN231E04 provides four configurable analog blocks, each containing eight capacitors, two operational amplifiers, and a comparator. Using switched-capacitor techniques and programmable interconnects, it can connect these resources to approximate[1] a range of common analog subcircuits with variable resistive/capacitive elements. The AN231E04 operates on 3.3V differential analog signals up to ~1 MHz in frequency.

Scanalog is capable of programming the same high-level modules as the AnadigmDesigner software, but can continuously update the circuit topology and parameters of the FPAA in real time as the user adjusts their design. After the user makes a modification to the circuit, the hardware reflects the corresponding change within a few milliseconds.

We summarize the continuous programming of the FPAA as follows: Scanalog internally represents the dataflow model of the GUI using a graph structure. Each module in the GUI corresponds to a known subcircuit implementing the given functionality, and for any change in the interface, Scanalog solves the governing equations of the subcircuit to choose component values yielding the desired parameters of the module. A resource allocation stage then reassigns fixed components of the FPAA for each subcircuit and stores in a graph the connections which must be made between components. Finally, bytes are flashed over serial to the appropriate configuration RAM of the FPAA to reset the board with these connections (i.e. switch box paths) and components values (i.e. switched capacitor frequencies). Because Anadigm does not release the memory map describing which memory locations and byte values control which aspects of the circuit, implementation of this final programming stage required reverse engineering of the mapping through repeated test circuit generation and comparison.

[1]We say "approximate" because the switched-capacitor implementation of the AN231E04 involves a certain frequency-dependent loss of precision over fixed components.

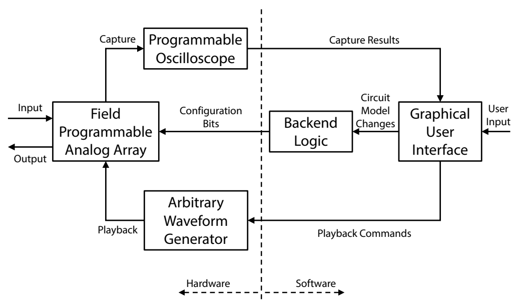


Figure 7. Functional diagram of Scanalog and the communications between its components.

Data Capture and Visualization

Scanalog uses two BitScope Micro BS05 USB Programmable Oscilloscopes [11] - one to capture and record signals, and another to generate signals for playback. When sampling a high-frequency signal, the oscilloscope must capture data faster than it can stream over USB. Thus, Scanalog captures "traces" (fixed-duration captures from a given signal) and updates visualizations within the interface between successive traces. The BS05 is capable of 20 MS/s analog capture and can store up to 12288 samples for a given trace. Scanalog allows the user to adjust the sampling rate and sample size of the oscilloscope to yield an appropriate window size for their signal (Figure 2.I). As a generator, the BS05 can output arbitrary waveforms composed of up to 1024 interpolated points with voltages ranging from 0 to 3.3 V.

To implement this real-time scanning functionality, the dedicated oscilloscope connects to an arbitrary output pin on the FPAA. Under the hood, while the user designs, Scanalog rapidly configures connections to this output pin from each module in the current circuit, triggering a capture for each module and updating the corresponding visualization before repeating. This process is invisible to the user, and runs in the background as they design.

Currently, a single atomic capture (reprogram a new connection to the oscilloscope, perform capture, read data and update visualizations) takes roughly 300 ms of overhead plus the duration of the specified capture. The equates to a ~3 Hz refresh rate between successive captures on a single module. Notably, because we choose to use a single oscilloscope and time-multiplex over all circuit nodes, the refresh rate on each module decreases linearly with the number of modules enabled for active visualization.

At any time, however, the user can choose to switch from autoscanning into a *manual probing mode* and use the oscilloscope as a higher-refresh capture device for any module of interest (Figure 2.G). Because in this mode the system does not automatically reconfigure to capture new targets, the capture overhead takes 150 ms, equating to a refresh rate of ~6 Hz for most signals. This is useful when fine tuning parameters where high-fidelity monitoring of the output is important. By default, the system updates all visualizations as soon as they are available. However, the user can also pause this view and manually record a signal from any module output.

Playback

With traces recorded, Scanalog can program the waveform generator to "replay" the signal back to the FPAA. This requires downsampling the signal to fit within the 1024 point buffer of the generator, then scaling and offsetting the voltages values to fit within the generator's positive output range. A custom amplification circuit soldered onto the FPAA inverts the scaling and offsetting on the generated signal to output the originally captured signal to an customizable input pin.

Tests and Assertions

Scanalog also uses data from captured signals to run its global debugging tests, assertions, and unit tests. Each capture is represented as an array of voltage values, and each module stores the most recent trace for use with tests. Global debugging tests scan the most recent data for each module, whereas assertion tests run on every data update for its connected module. Finally, Scanalog implements unit tests by replaying recorded signals through a circuit configured with assertions and reporting the results of those tests.

```
/* Returns true iff the signal has voltages below ground */
function hasNegativeVoltages(data, m)
{
    for (var i = 0; i < data.length; i++)
    {
        if (data[i] < 0)
        {
            return true;
        }
    }
    return false;
}
```

Figure 8. A basic user-written test which alerts when a signal has negative voltages. Test functions are called with a module object ('m') containing the module parameters, and an array of voltage values ('data') representing the latest trace for that module.

To enable user-defined tests, Scanalog stores all tests as string-represented function objects which can be edited in text within the application. To add a new test, the user provides a name for the test, adds an explanation to report on test failure, and then edits an skeleton code of an example test to implemented out the desired check. Tests can access the array of voltages captured in the most recent trace, as well as parameters of the module on which it runs. A basic example of a user-defined test is shown in Figure 8.

EXAMPLE RESULTS

We created three interactive systems to demonstrate Scanalog's potential to assist with a range of analog design tasks. The first, described in the Walkthrough section (Figure 3), is a 2-lead EKG monitor capable of visualizing the electrical activity in an individual's heart.

The second application (Figure 9) is an interactive audio mixer capable of filtering, crossfading, and volume-adjusting tracks in real-time. The user first connects the signal lines from an audio source (e.g. the 3.5 mm headphone jack of an MP3 player) to the input of the FPAA, and then connects a speaker to the output. Because the audio signal is an analog waveform, Scanalog can manipulate the sound directly without sampling. We implement crossfading by connecting multiple sound sources to a summing stage, which can add their signals together with adjustable coefficients. The combined output of the summing stage passes first through a low-pass and then a high-pass filter. At each stage, the user can adjust the gain (volume) of the signal and the frequency ranges which pass through.

The third application (Figure 1) shows the conditioning of an analog sensor to output a digital signal. We use a QRB1134 IR photoelectric sensor, which detects proximity and reflectance of objects. The sensor has an LED which emits infrared light, coupled with a photoresistive element tuned to the same wavelength. As objects come into proximity and reflect light back into the receiver, the resistance of the sensor changes. Our application configures the sensor for proximity sensing (to

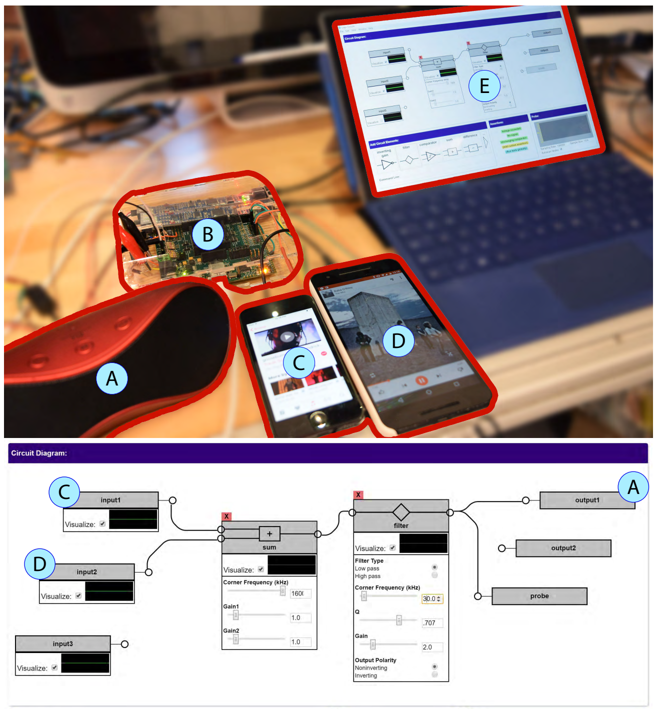


Figure 9. Real-time audio crossfading by summing analog signals. Top: The setup used, including a speaker (A), two audio sources (C and D), and the Scanalog hardware (B) and software (E). Bottom: The audio circuit, consisting of summing and filtering stages.

detect user input), as well as to sense changes in material with differing reflectances (as a sensing device for robots). The circuit consists of a preliminary gain stage (to increase the depolarization of the signal in response to an object), followed by a variable reference comparator to output a digital high or low based on a tunable threshold. As the sensor requires more current than the FPAA can source, we also utilize a separate power circuit constructed on a breadboard.

INFORMAL EVALUATION

To solicit feedback and further understand how users are able to make use of Scanalog, we invited eight university students to explore its features and to complete a circuit design task in an informal evaluation. All students had some previous exposure to prototyping circuits on a breadboard, but none were experts in analog circuit design. We walked each user through Scanalog's features and the basic workflow, then handed off control for the design task. The task involved creating a proximity-sensing switch using the IR photoelectric sensor described in the previous section. Specifically, the user's circuit needed to output a digital high value when the user covered the sensor with their finger and a digital low otherwise. The instructions consisted solely of this specification, and the user had access only to the Scanalog interface to complete the task.

All participants were able to complete the task successfully, and no participant took longer than seven minutes to reach a working solution. While some students deduced the necessary modules right away, others took an exploratory route through several incorrect or near-correct approaches. Likert responses

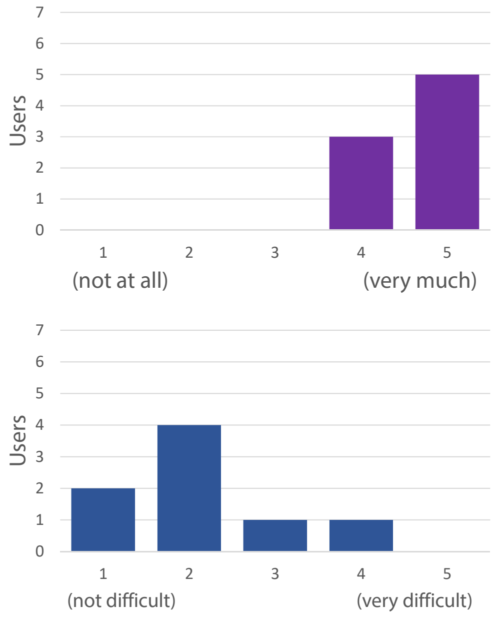


Figure 10. Likert responses to the questions, "Did you like the system?" (top), and "How hard was it to complete the task?" (bottom).

to a post-task survey revealed that all users enjoyed using Scanalog, while most (but not all) users reported completing the task without much difficulty (Figure 10).

When asked about the aspects of Scanalog that they found most useful, all participants mentioned the real-time visualization of internal signals, and six of eight mentioned the ability to iterate quickly with real-time circuit modifications. Participants also spoke to how these features might assist them in their own projects and work habits. For example, one user described how Scanalog's ability to quickly tune each component while actively visualizing the others would have helped her to combine filtering with gain stages in conditioning an accelerometer for a design project: *"Filtering is very hard, because all you see is a noisy signal, and it's a pain to try out a bunch of different filters and try to judge if it's getting better or worse. Being able to just see the signal and very quickly mess with the parameters would be so helpful."*

In follow-up interviews, users compared Scanalog to the tools that they typically used, reporting a greater degree of immediacy: *"It's great that circuits are physical, but there's nothing that you can detect with your own senses without having to probe or use these other instruments. The closer I can get to what's happening in the circuit, the easier it is for me to understand what's going on."* As compared to their conventional tools (hand drawn schematics, graphical circuit design tools, physical breadboarding, oscilloscopes, and multimeters), users unanimously felt that Scanalog made the completion of the task faster and simpler.

Users also suggested a number of improvements to Scanalog, primarily in the form of feature requests. Among these requests were: higher level interface tools like an "undo button," saving and loading of custom circuits, and the ability to edit the underlying code of assertions in the workspace. Two participants remarked that the ability to directly overlay live visualizations from multiple modules would enable them to more easily compare the differences between signals.

LIMITATIONS

Despite its utility, the current version of Scanalog still has limitations which pose opportunities for future work.

Many such limitations are inherent to the underlying hardware chosen for our implementation. The AN231E04 FPAA has a finite set of resources for configuring modules limited in both type and quantity, and it cannot operate on high frequency signals above ~1 MHz. For applications requiring great precision, idiosyncrasies resulting from its switched-capacitor behavior can cause noticeable deviations from a version of the same circuit using true resistive elements. Similarly, the oscilloscope and generator both have limits in the frequencies, durations, and precisions of signals with which they can work. However, Scanalog can easily be implemented on different platforms with different performance characteristics.

The refresh rate of Scanalog's visualizations is limited due to the time-multiplexing of a single oscilloscope over multiple modules in a circuit, as described in the Implementation section. As a result, capturing long signals on the order of seconds can noticeably delay updates for the rest of the circuit. While the current refresh rate can be improved with software optimization, applications requiring high refresh rate would benefit from the use of multiple oscilloscopes working in parallel to provide continuous capture at high bandwidth.

Finally, while the high-level modules currently implemented on Scanalog allow for the manipulation of signals in a number of flexible ways, they do not cover all possible analog circuits, and more expert users may desire to work in a lower-level fashion with raw analog components. Future versions of Scanalog can not only offer more modules, but can expose the raw resources of the board, affording expert users the flexibility to combine op-amps, comparators, and switched capacitors into arbitrary configurations.

FUTURE DIRECTIONS AND CONCLUSION

Scanalog addresses the difficulties of analog circuit design by offering users direct manipulation of circuit parameters, immediate feedback of circuit behaviors, and automated debugging features. Early feedback is encouraging to show that these techniques enable users to more rapidly and iteratively solve analog design tasks.

Currently, FPAAs and related devices are marketed towards professional hardware designers, allowing them to more quickly test designs without manual fabrication. However, with an accessible tool that facilitates rapid prototyping and debugging, we hope to widen the application space of FPAAs to students, makers, and even users from non-engineering domains. For example, direct manipulation of analog stages can aid artists or designers of musical interfaces in rapidly exploring a design space, while assertions and unit tests can be useful in robotics when tuning sensors and actuators in a control flow system. We intend to deploy Scanalog in the wild with users of varied skills levels and to study in detail how their workflows change using its various supportive features.

Further, research can apply the core principles of Scanalog — instrumentation to enable direct manipulation and immediate feedback — more generally across other domains, such as medicine or product design. Our work will continue to explore such opportunities moving forward, as we believe that these techniques can be invaluable in designing and understanding the behaviors of complex systems.

REFERENCES

1. Yoh Akiyama and Homei Miyashita. 2014. Projectron Mapping: The Exercise and Extension of Augmented Workspaces for Learning Electronic Modeling Through Projection Mapping. In *Proceedings of the Adjunct Publication of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST'14 Adjunct)*. ACM, New York, NY, USA, 57–58. DOI: http://dx.doi.org/10.1145/2658779.2659113

2. Altium. 2017. Altium Designer 17 Overview. (2017). http://www.altium.com/altium-designer/overview.

3. Anadigm. 2017a. AN231E04 dpASP. (2017). http://anadigm.com/an231e04.asp.

4. Anadigm. 2017b. Anadigm Designer2 Software. (2017). http://anadigm.com/anadigmdesigner2.asp.

5. Autodesk. 2017. EAGLE PCB Design and Schematic Software. (2017). http://www.autodesk.com/products/eagle/overview.

6. Ayah Bdeir. 2009. Electronics As Material: LittleBits. In *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction (TEI '09)*. ACM, New York, NY, USA, 397–400. DOI: http://dx.doi.org/10.1145/1517664.1517743

7. Tracey Booth, Simone Stumpf, Jon Bird, and Sara Jones. 2016. Crossed Wires: Investigating the Problems of End-User Developers in a Physical Computing Task. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 3485–3497. DOI: http://dx.doi.org/10.1145/2858036.2858533

8. Joshua Chan, Tarun Pondicherry, and Paulo Blikstein. 2013. LightUp: An Augmented, Learning Platform for Electronics. In *Proceedings of the 12th International Conference on Interaction Design and Children (IDC '13)*. ACM, New York, NY, USA, 491–494. DOI: http://dx.doi.org/10.1145/2485760.2485812

9. Bettina Conradi, Verena Lerch, Martin Hommer, Robert Kowalski, Ioanna Vletsou, and Heinrich Hussmann. 2011. Flow of Electrons: An Augmented Workspace for Learning Physical Computing Experientially. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces (ITS '11)*. ACM, New York, NY, USA, 182–191. DOI: http://dx.doi.org/10.1145/2076354.2076389

10. Cypress Semiconductor Corporation. 2017. PSoC Software. (2017). http://www.cypress.com/products/psoc-software.

11. BitScope Designs. 2017. BitScope Micro Model 5. (2017). http://www.bitscope.com/product/BS05/.

12. 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 (UIST '16)*. ACM, New York, NY, USA, 677–686. DOI: http://dx.doi.org/10.1145/2984511.2984566

13. Adam Fourney and Michael Terry. 2012. PICL: Portable In-circuit Learner. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology (UIST '12)*. ACM, New York, NY, USA, 569–578. DOI: http://dx.doi.org/10.1145/2380116.2380188

14. Saul Greenberg and Chester Fitchett. 2001. Phidgets: Easy Development of Physical Interfaces Through Physical Widgets. In *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology (UIST '01)*. ACM, New York, NY, USA, 209–218. DOI:http://dx.doi.org/10.1145/502348.502388

15. Björn Hartmann, Leith Abdulla, Manas Mittal, and Scott R. Klemmer. 2007. Authoring Sensor-based Interactions by Demonstration with Direct Manipulation and Pattern Recognition. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '07)*. ACM, New York, NY, USA, 145–154. DOI: http://dx.doi.org/10.1145/1240624.1240646

16. National Instruments. 2017. LabVIEW System Design Software. (2017). http://www.ni.com/labview/.

17. 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 (TEI '09)*. ACM, New York, NY, USA, 351–358. DOI: http://dx.doi.org/10.1145/1517664.1517735

18. Johnny C. Lee, Daniel Avrahami, Scott E. Hudson, Jodi Forlizzi, Paul H. Dietz, and Darren Leigh. 2004. The Calder Toolkit: Wired and Wireless Components for Rapidly Prototyping Interactive Devices. In *Proceedings of the 5th Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques (DIS '04)*. ACM, New York, NY, USA, 167–175. DOI: http://dx.doi.org/10.1145/1013115.1013139

19. 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, New York, NY, USA, 1270–1281. DOI: http://dx.doi.org/10.1145/2901790.2901833

20. Laurence William Nagel and Donald O Pederson. 1973. *SPICE: Simulation program with integrated circuit emphasis*. Electronics Research Laboratory, College of Engineering, University of California.

21. Yoichi Ochiai. 2010. The Visible Electricity Device: Visible Breadboard. In *ACM SIGGRAPH 2010 Posters (SIGGRAPH '10)*. ACM, New York, NY, USA, Article 98, 1 pages. DOI: http://dx.doi.org/10.1145/1836845.1836950

22. Janneke Verhaegh, Willem Fontijn, and Jettie Hoonhout. 2007. TagTiles: Optimal Challenge in Educational Electronics. In *Proceedings of the 1st International Conference on Tangible and Embedded Interaction (TEI '07)*. ACM, New York, NY, USA, 187–190. DOI: http://dx.doi.org/10.1145/1226969.1227008

23. Bret Victor. 2013. Media for thinking the unthinkable. *Vimeo, May* (2013).

24. Nicolas Villar, James Scott, Steve Hodges, Kerry Hammil, and Colin Miller. 2012. *.NET Gadgeteer: A Platform for Custom Devices*. Springer Berlin Heidelberg, Berlin, Heidelberg, 216–233. DOI: http://dx.doi.org/10.1007/978-3-642-31205-2_14

25. 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 (UIST '16)*. ACM, New York, NY, USA, 687–695. DOI: http://dx.doi.org/10.1145/2984511.2984527