Graduate Research Statement: Software-Defined Hardware

Branden Ghena <brghena@umich.edu>

There has been a revolution in the development of embedded systems. Enabled by inexpensive, accessible platforms like the Arduino and Raspberry Pi, creating new devices is no longer the exclusive domain of the expert engineer. For the first time, artists and amateurs alike are solving difficult problems and implementing innovative ideas. But a high bar still exists for translating ideas into reality. For example, in order to sense the ambient temperature, a number of questions must first be answered: What method of sensing is desired? What I/O interface should be used? At what voltage is the system operating?

Even those well versed in software design can be intimidated by the sheer number of options available to them and choices that must be made. The reality though, is that many users simply want to know the temperature, they do not care how it is sensed. My work proposes to bridge this gap. My **hypothesis** is that embedded code either already includes enough information to implicitly define the hardware that it requires, or could be trivially augmented to do so.

Goal of Proposed Research. The goal of this project is to demonstrate that it is possible to generate a design for an embedded system's hardware from its software, and to develop a methodology for doing so. My vision for an eventual use of this research would be a website where users could upload embedded code, and it would determine which devices are required, lay out a PCB including them, and give users the option to have it fabricated and shipped. Instead of looking at a list of hardware plugins and selecting applications that use them, this tool would enable users to write application software, and automatically have hardware to support it. The choices of which components best fit the embedded system could be gleaned from the software itself.

Research Plan. In order to evaluate my hypothesis, a proof of concept is necessary. The first step is to define the scope of the project. While the technique should ultimately be extensible to any platform, limiting initial testing to one selection will eliminate complexities from the project while still revealing useful information about its fundamental viability. Also limited will be the number of connectable devices. For each supported device, a database cataloging device characteristics, software interface, and schematics will be needed. This is a non-trivial amount of effort, but only needs to be applied to small number of representative devices initially.

Two important factors are the types of requirements each device can fulfill and the additional requirements that would be added to the system through its use. For example, using a specific temperature sensor will fulfill a system's need for ambient temperature knowledge, but place additional requirements that it must communicate via a certain bus and run at a particular voltage.

The next major step is to create a system for automatically parsing requirements from software. This involves both being able to recognize the need for a component and deciding which factors must be taken into account when selecting it. Requirements can be levied on the device based on how its data is utilized. For example, library calls can be used to detect that a part is required,

while the use of data returned from that library call in a loop or data transmission function can describe the sampling frequency required for the device.

A critical aspect of this step is to identify the aspects of the system that cannot be determined from the code itself. These aspects will have to be specified through augmentations to the already existing code if they are requirements of the system. Otherwise, if they are simply options to be chosen between, such as I/O interface, heuristics can select based on other characteristics of the part like price, pin count, or power draw.

Once requirements are set, designing the embedded system can be treated as a resource allocation problem. Various heuristics can be applied in order to minimize system costs when selecting between available components. Connecting the individual device schematics to the chosen platform can generate an overall system design.

A successful result from this research will be the capability to automatically generate a hardware schematic from embedded software. In the future, this technique could be extended to create a tool that would automatically generate a PCB and parts list for a given piece of code.

Previous Work. The advent of hackerspaces and amateur embedded systems like the Arduino and Microsoft Gadgeteer has begun the push towards enabling a new class of innovators. These systems are easily extensible through the connection of various hardware modules. This is a step in the right direction, allowing anyone to utilize hardware simply by purchasing an adapter.

Another methodology similar to this one is high-level synthesis. It focuses on generating FPGA logic implementing the behavior of software, whereas this research focuses on generating an embedded system design to complement the software.

Intellectual Merit. The difficulty in creating software-defined hardware is rooted in specifications. In current practice, a domain expert must decide that based on the requirements, *this* handful of chips will best satisfy the demands of the system. The challenge comes from properly creating and understanding specifications. This research also explores several high-level questions such as how hardware artifacts may be best represented in software to support composition, and how to capture non-functional requirements such as lifetime or reliability.

Broader Impacts. Software-defined hardware has the potential to change the focus of the embedded field. The technological expertise and hard work that went into creating an iPod is not what makes it a useful product. It is the application of the device that really drives its success. The focus of the field should be on discovering unique applications, not resolving mundane technical plumbing of the hardware. A system to generate hardware out of software is a first step towards application-driven design accessible to the masses.

The creation of a system for hardware generation would also enable an entirely new subset of the population to innovate and create. Whereas today the technical know-how necessary to create new embedded systems only comes from years of training and experience, this system would provide the ability for a much larger audience to enter the role of creator.