Furthering the Development of a Digital Microfluidic Platform

Andrew F. Dreher

April 27, 2005

Abstract

Digital microfluidic systems are an emerging implementation of lab-on-a-chip systems based upon the manipulation of discrete droplets. Due to the volume of the fluid droplets, massive integration and highly sensitive tests, difficult to realize using other technologies, are possible. It is imperative to develop the technology to the point were it can be deployed for use by researchers and other professionals whose background does not include electrical and computer engineering. This project involved furthering the development of a microfluidic platform and studying such topics as chemical compatibility with and user operation of the platform. Updated control software has been written based on object oriented paradigms to provide a better user experience and to evolve as the microfluidic platform develops. The current version of the control software allows for higher level operation of the device so that the user need not program using machine code. In the future, even more sophisticated system software will allow for easier programming and deployment by end users.

1 Introduction

Digital microfluidic systems are lab-on-a-chip systems based upon the manipulation of discrete droplets using the concept of electrowetting. The current implementation of the digital microfluidc platform at Duke allow for dispensing, mixing, combining and splitting – the fundamental building blocks of the digital microfluidic system. Unfortunately, in its current realization, these tasks require careful supervision by a trained operator. This paper will present some of the research and development which will hopefully, lead to a digital microfluidic platform which will be able to be used with less intervention.

Digital microfluidics is a rapidly emerging technology in the lab-on-a-chip universe, and as an emerging technology, there are many fundamental issues to be solved, from the design and control of processes to chemical compatibility and manufacturing methods. A broad range of problems related to the development of a microfluidics platform will be illustrated in this paper. For reference, the term *platform*, will be used to refer to a system, the combination of the hardware and the software, upon which desired routines, products and tests may be developed. Bringing microfluidics from an emerging technology to a platform will require a considerable amount of research into the techniques and control of such a system. Having a well designed and developed platform should enable technical users whose background is outside electrical and computer engineering to successfully leverage microfluidic technology.

2 Ammonia Analysis

This project started in the Fall 2004 as a way of scanning and analysing the amount of ammonia impacted on the surface of a digial microfluidic chip. Scanning was to be done after impact. Later that process would conceivably be implemented using coplanar technology so that scanning and impact could occur without special preparation. The analysis of ammonia in air samples are of great interest to researchers studying the atmosphere, and this system was to be designed to aide their reserach.

Chemical compatibility is a critical issue for the digital microfluidics platform. It is necessary to find chemical reactions which produce the desired products and can be reacted on the scale of the microfluidics platform; the chemicals must also be able to be moved by the electrowetting process and not react with the teflon coating, the substrate, should it become exposed, or the oil. For the reagents to be compatible with the electrowetting process, the reagents must be able to be moved by a change in the electrical potential and must separate from the oil.

Ammonia was to be detected on chip using a wet chemistry process; the normal method of gas diffusion was not acceptable for a lab-on-a-chip process and was thus discarded from consideration. There are two well established, available wet-chemistry processes, the Nessler and Berthelot tests. Both of these tests satisfied the requirement of being able to be monitored by a photometer; additionally, both should have been able to detect ammonia in the concentration of ng/L (the target concentration). Unfortunately, the reagents for the Nessler and Berthelot tests were unacceptable for use in our lab: both the Nessler and Berthelot reagents are both volatile, health hazards, and react with materials commonly used in the construction of the hardware. The health hazard was an important deciding factor in not pursuing further laboratory work. The reagents for both tests can be absorbed through the skin and are neurological toxins. The Nessler reagent was that it reacts with glass due to its high alkali content, and since the digital microfluidics platform at Duke mainly uses chips fabricated on glass substrates and a glass top plate. The problem with the Berthelot test, is that the reagent reacts with plastic, another common component of our current microfluidic system.

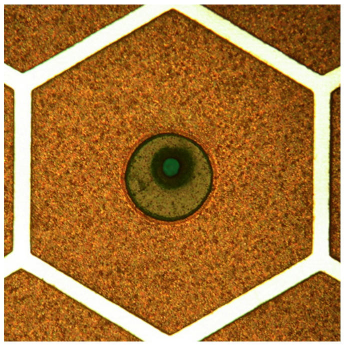

Figure 1: Photograph of a chip fabricated in PCB technology under combined front and back lighting

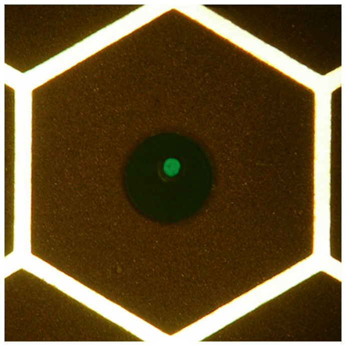

Figure 2: Photograph of a chip fabricated in PCB technology under backlighting only

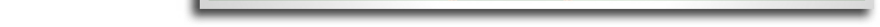

Figure 3: Structure of a chip realized in PCB technology

3 Profiling and imaging of chips fabricated in printed circuit board technology

Moving from a single metal layer on a glass substrate to printed circuit board technology allowed for backside contacts which should make the design and construction of arrays and other larger scale integration layouts easier. Using printed circuit board, PCB, technology is also more economical and has a faster turn around time than chips fabricated using metal layers on glass. Chips fabricated in PCB technology use copper over resin with a minimum geometry metaled through hole for the backside contact. The hole was then filled with solder mask to prevent the droplet from being siphoned to the backside. An example of the filled hole in a PCB technology chip is shown in Figure 1. This image is double exposure using both front and back lighting. The soldermask can be seen overlaying the copper in the center of the hexagonal pad; the through hole can be seen as the lighter green region inside the larger soldermask deposition. The profile of a chip realized using PCB technology is shown in Figure 3. Using a profilometer, the copper pads were found to be approximately 250,000 thick and the solder mask layer was found to be approximately 265,000 thick. The center of the solder mask was approximately 400,000 below the maximum level of the solder mask. It is important to note that the minimum level of the solder mask is below the surface level of the copper. The potential for the irregularity in the surface to disrupt droplet motion requires further study. Currently, PCB based chips are being used successfully in the laboratory; however, they require large activation voltages. The performance of PCB based chips requires further consideration to determine the performance.

Measurements were taken at appoximately 10 locations on a single chip for reference. Since the measured values were within the specified values of the commercial process, further analysis was deemed unnecessary at the current time.

4 Microfluidic System Software

4.1 Introduction

The system control software provides a better interface to the user and makes it possible to run a routine on the digital microfluidic chips. Ideally, the system software takes the process information from the user and permits the routine to be executed on a variety of different hardware implementations. This system software was designed using object oriented, design principles to provide a starting point from which advanced system control software may be designed and written. The eventual goal is to be able to graphically enter a control chart (similar to ASM charts used in hardware design or UML diagrams used in software design) and allow the software to layout and schedule the routine. The control software is a work in progress, however, because it is not possible to jump from machine code directly to a sophisticated, automated tool. Not all of the scheduling or layout mechanisms and heuristics are known; they must be discovered. The system control software must therefore be able to evolve as new information and techniques are discovered and implemented. Using object oriented design paradigms should allow for this flexibility.

Since moving from a fully manual, machine code implementation to a fully automated implementation was not possible, the project started by providing a graphical interface by which the device may be programmed. Since the connections and states are held logically, not physically, a change in the interface to the device does not require reprogramming the device as it did in the previous generation of control software. It is also possible to easily ”see” the layout and activation states as opposed to programming ”1” and ”0” into a giant array. Although it still lacks almost all automation, graphical programming should make the designs less error prone and be faster to complete. The next step in the evolution of the software is to provide a manual layout and the necessary process flow charts. It will then be possible to use automated schedulers, such as modified versions the popular A* or A*alpha schedulers used in game programming. It will then be possible to make the final leap to a fully automated system.

4.2 Design

The design of the system control software is very important because the software must be usable both as a mostly manual system and as the basis for a fully automated system. The software was designed in Java to provide maximum flexibility in platform selection. As the platform moves from a completely manual solution requiring frequent user intervention to a fully automated solution, complete with feedback and adjustment, the desired platform is likely to switch from a graphical, desktop application to an embedded application. Programming the software in Java will hopefully allow this flexibility with maximal code reuse. This projected evolution of the control software makes the underlying design rather challenging.

For the program and design of the system, a basic layout was established. Future features and revisions will likely follow the eXtreme Programming, henceforth XP, paradigm where releases are frequent and changes are minor. This is a central philosophy of the design of the Linux operating system and many open source and academic projects. The code will likely undergo many ”refactorings” to improve the quality of the code, but the basic structure should remain intact. The basic layout of the software, shown in Figure 4 consists of a central controller interacting with several object controllers. The object controllers each contain several other object to perform the desired functionality. The system relies heavily on the Inheritance and Decorator patterns. The book *Design Patterns: Elements of Reusable Object-Oriented Software* is a great resource for further investigation of these patterns [4].

The software is realized using the Model-View-Controller, MVC, design pattern. The graphical user interface is rendered by the graphical controller. All of the graphics code is contained in this and related classes. By restricting the graphical view to the graphical block, the software may be accessed using a non-graphical controller, shipped without a graphical control (for size / performance considerations), and updated easily. The graphical controller relies on four sub-blocks: the control window, the inspector, the groups window and the

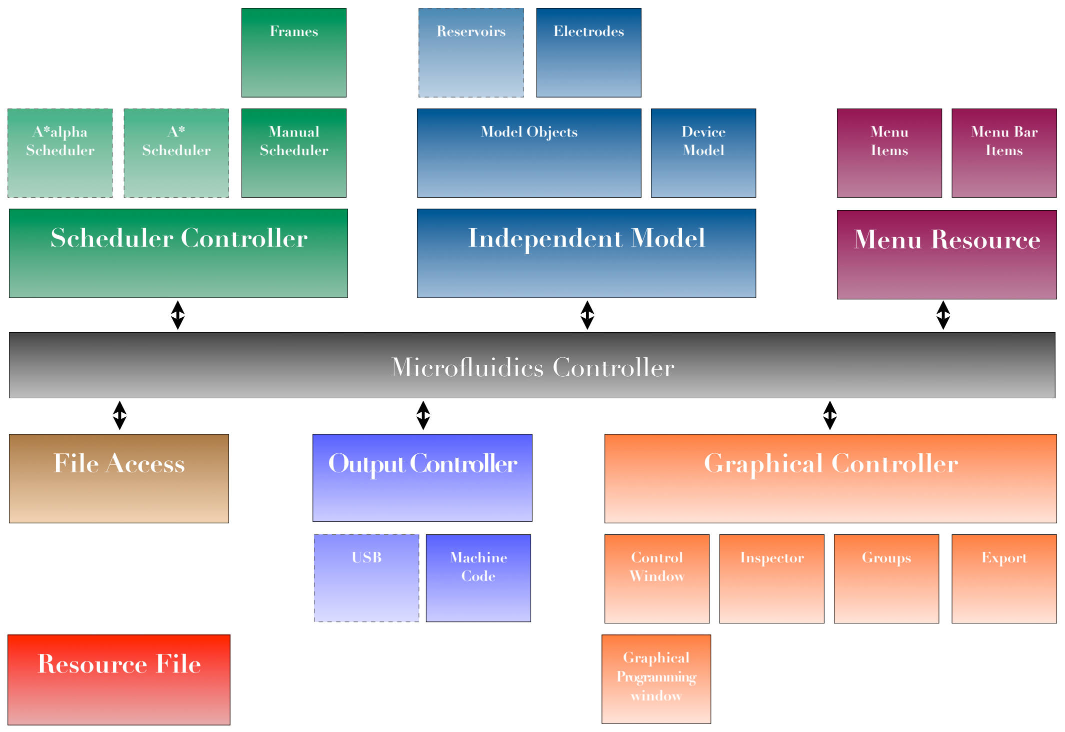

Figure 4: Complete block diagram of the microfluidic control software

export dialog. The control window is the primary interface to the software. It contains the graphical programming mechanism and visualization of the chip being programmed. The inspector window provides information about the model elements and allows editing the name of the element and group to which the element belongs. The inspector also provides information such as the type of element under consideration. It will eventually provide more information, such some statistics about the usage of an element, as well. The groups window allows editing of the group name and the output pin to which the group is electrically connected. Currently, only electrically connected groups are supported, but future versions of the software will support groups based soley on a logical association as well. Finally, the export dialog is responsible for providing a reasonable interface for exporting the design into a variety of formats.

The user interface was a major design consideration for this project. Despite still being quite low level and manually programmed, it was hoped that the software would be easy to use and accelerate writing routines for the platform. Careful presentation of data to the user was therefore important.

Every frame is programmed individually, but since frames are run sequentially, establishing a relationship between subsequent frames is important. To emphasize that droplets do not move instantaneously from an electrode to an adjacent electrode activated on the subsequent frame instantaneously, a graphically represented "recently active" state was included in the manual scheduler. In order to keep the visualization as simple as possible,

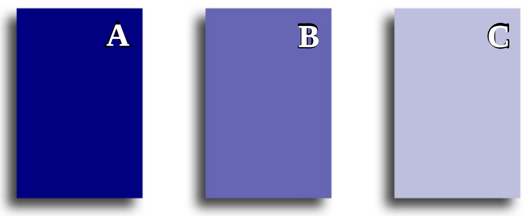

Figure 5: Example of Alpha Channel effects

color was reserved for static situations, such as differentiating the types of objects displayed. The alpha channel (transparency) was selected as the methodology for emphasizing the state of an electrode. Over most backgrounds, the change in the alpha channel will result in a value change, while the hue remains constant. This should aid in identifying not only the state of an object, but also it's function. This scheme for rapid identification should make programming in this environment rather intuitive. To further emphasize the graphical design, more saturated colors are used for active states, whereas a less saturated color denotes an inactive state. In the current version of the software, an alpha value of 1.0 was used for active elements (Figure 5A), 0.25 for inactive elements (Figure 5B) and 0.6 for recently active elements (Figure 5C). The color saturation scheme should also help to draw the user's attention to active states, but allow passive states to fall more into the background.

Because this software was designed with the Apple Macintosh as the development platform, special attention was paid to make the interface as close to the native design as possible. In so far as was reasonable, the Apple Human Interface Guidelines [1] were followed. Since Solaris and Macintosh OS X have different menu expectations (such as "Quit" under the Application menu on Macintosh OS X versus "Exit..." under the file menu on Solaris) differing versions of the menus are loaded based upon the target platform. Although there are some differing menus, many items remain the same and are thus reused. To control management of all the menu / menu bar items, a single menu resource location was created. This was done using a singleton design pattern [4] and relates naturally to the macintosh's single menu bar design. In Figure 4 the resource center is shown as *Menu Resource*. Using the singleton design pattern also ensures that menus such as "open", "save" and "quit" remain consistent, both in appearance and function, throughout the application.

To allow for maximum flexibility, data is stored in a model irrespective of the device on which it is implemented. In fact, the geometry of the device is also not taken into consideration in the device independent model (Independent model in Figure 4). The model stores the data with logical connections. When an automated scheduler is employed, such as one based upon A*, storing the elements as nodes in a graph makes sense. Edges in the graph will represent adjacent cells and will be based upon the geometry of the circuit. With a manual scheduler, none of this detail is important, but the groundwork needs to be laid for future development. The representation of the device, in the form of electrical output, is held in a separate model. Translation from the logical to the electrical model occurs only as needed. Interacting with the logical model is an important part of manual programming. To facilitate improved programability, it will be possible to extend the current electrical groupings into purely logical (non-physical) groupings as well. While a model element currently may belong to only one group, this need not be the case.

The scheduler system is accessed though a scheduler control block which therefore allows many different schedulers to be accessed identically. It may also be possible in the future to use the dynamic class loading feature in Java to load schedulers as needed, thus providing more value to the user while reducing the system overhead. Currently, only the manual scheduler is accommodated. As mentioned earlier, frames as a way of storing the manually written script. All of the frames are stored as in internal model to the manual scheduler. This ensures a level of separation between the model and the scheduler. In future, automated, schedulers, different underlying schedule models will be needed to correctly guide droplets. This encapsulation ensure that OOP principles are followed. The scheduler system interacts closely with the output system. The two systems, however, should remain distinct for maximum flexibility. Though not currently utilized, a system based upon the command design pattern has been devised for linking the output to the scheduler. When output is requested, the scheduler will provide a current *command* to be decoded and executed by the output block.

Files are loaded though the file access block. By restricting file access to a single source, strict controls can be placed on the use of files. Currently, files are loaded in unencrypted, plain text XML. Eventually, however, it is conceivable that a digital rights management, DRM, type system may be included to protect sensitive content. The full layout of a chip or the design of a testing method may be proprietary information, and may need to be protected as intellectual property. Providing this option increases the flexibility of the software design.

4.3 Interoperability

The control software is but one component used in the design and implementation of the microfluidic platform, and it is imperative that the control software be able to interact with the other tools. With the increasing globalization of projects, it is also important to be able to bridge languages so that all make work on a common piece of software. In this project, the power of XML and Java is leveraged to achieve these interoperability goals.

Most of the layout for the microfluidic chips is currently performed using tools from Mentor Graphics. In the future, it would be highly advantageous to convert designs from Mentor Graphics into a format which will be able to be used by the control software. By choosing industry standard XML as the basis for our file format, this ensures that the tools can be written with minimal knowledge of the control software. These files will therefore be more portable and better supported than if a proprietary format were chosen. Choosing XML as the basis for our format also allows our control software to leverage the development of other projects using XML, thus saving time and avoiding errors in loading and parsing our files.

Java was chosen mainly because of it's ability to run in many different computing environments with a minimal change to the source code. Using Java, however, does provide many other benefits. Because of it's built in support for resource files and unicode, many different languages, including those based on non-Roman alphabets or characters, may be supported. Using a resource file for every string in the program allows changes in language to be performed solely on the resource file. This also provides the assurance that all of the strings will be translated without having to search through the source code. The impact on performance using this method should be minimal on modern systems.

5 Conclusion

There remains a lot of work to be done to bring microfluidics out of the research setting and to establish a microfluidic platform. As mentioned previously, there are issues related to chemical compatibility; design, manufacturing and performance of the chips; and many control software related issues. Many of the most challenging issues will need to be researched before a ”turn-key” like solution will be available, but this is the ultimate goal. The ability to perform tests with previously unimagined accuracy and at high levels of integration will benefit all. Placing this technology in the hands of professionals who will be able to develop the tests and solutions is paramount.

Everyone working in microfluidics must be part electrical engineer and part chemist. Most of the techniques and research being performed in the field is directly related to the tests being carried out on the current platform. It is imperative that the range of people who have access to the technology be increased. Wonderful research is currently being performed, but the full ability of the technology will only be realized when placed into the hands of researchers in the field under study. When I began this project, the goal was to analyze the ammonia content present in air using the microfluidic platform. I am, however, not an environmental engineer, nor am I an advanced chemist. If, however, a reliable platform were developed, the environmental engineers and chemists could experiment with the platform themselves, probably providing more value and understanding to them. It was therefore my goal to further the development of such a platform by making the control software easier to use.

A fully automated micofluidic platform is still several years away. However, it is possible to take steps which will ease the development of those fully automated tools. Starting with software designed using object-oriented concepts provides the flexibility to adapt to developments in the tool while providing the current user with a better interface. The step from machine code to graphical programming is not wholly unlike the development of assembly from machine code. While both are low level and explicitly manage resources, assembly provides a platform for programming processors conforming to the same instruction set architecture (ISA). Likewise, programming the chips based on their geometry, not solely their electrical connections, allows any chip having the same geometry to be used by the routine, regardless of electrical connection to the computer.

The next step in the evolution of the control software is to move from explicitly controlling the electrodes to providing the software with a layout and a flow chart. The next step in the development of the control software will not be to move to a fully automated solution.

Rather, it will be based upon an automated scheduler following a flow chart specified by the user. The functional layout of the chip (mixing areas, pathways, etc.) will still be manually provided. This will be realized when an automated scheduler has been developed. Ding presented an architecture discussing the reconfigurable system in his thesis [3]. This type of layout could be used as the next stage in building automated tools, furthering the development of a digital microfluidic platform.

Acknowledgments

This project could not have been completed without the mentorship and advise of Dr. Richard B. Fair. Special thanks to Mr. Robert C. Duvall for his advice on the programming of the Java menu system.

References

[1] Apple Human Interface Guildlines. Technical report.

[2] Java 1.4 development for mac os x. Technical report.

[3] Jie Ding. *System Level Architectural Optimization of Semi-Reconfigurable Microfluidic System*. PhD thesis, Duke University, 2000.

[4] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Reading, Massachusetts, 1995.