IMP²

(INTELLIGENT METRONOME POWERED BY PSOC)

April 29, 2007

Brian Campbell
Michael Kabala
Justin Mirarck
Robert Utrup
IMP²

(INTELLIGENT METRONOME POWERED BY PSOC)

By

Brian Campbell
Michel Kabala
Justin Mirarck
Robert Utrup

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING IN
PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR EE402

ADvised BY

Dr. George Collins, Faculty Advisor
Dr. Cameron Moore, Project Advisor

APRIL 29, 2007
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Content</th>
<th>Page Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cover Page</td>
<td>i</td>
</tr>
<tr>
<td>Title Page</td>
<td>ii</td>
</tr>
<tr>
<td>Table of Contents</td>
<td>iii</td>
</tr>
<tr>
<td>List of Figures and Tables</td>
<td>v</td>
</tr>
<tr>
<td>Abstract</td>
<td>1</td>
</tr>
<tr>
<td>I. Introduction</td>
<td>2</td>
</tr>
<tr>
<td>II. Summary of Previous Work</td>
<td>3</td>
</tr>
<tr>
<td>III. Project Management</td>
<td>3</td>
</tr>
<tr>
<td>IV. What is the PSoC</td>
<td>6</td>
</tr>
<tr>
<td>V. The Metronome</td>
<td>13</td>
</tr>
<tr>
<td>VI. Software and the Modes</td>
<td>14</td>
</tr>
<tr>
<td>VII. Code Statistics</td>
<td>20</td>
</tr>
<tr>
<td>VIII. The Hardware</td>
<td>21</td>
</tr>
<tr>
<td>IX. Final Design</td>
<td>25</td>
</tr>
<tr>
<td>X. Design Challenges</td>
<td>28</td>
</tr>
<tr>
<td>XI. Budget</td>
<td>29</td>
</tr>
<tr>
<td>XII. Manufacturability and Marketability</td>
<td>32</td>
</tr>
<tr>
<td>XIII. Conclusion</td>
<td>34</td>
</tr>
<tr>
<td>XIV. Works Cited</td>
<td>35</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

Figure 1  PSoC Designer Device Editor
Figure 2  PSoC Designer Application Editor
Figure 3  Example Drum Output Waveform to PSoC
Figure 4  Button System for Changing Modes (Schematic)
Figure 5  Visual Circuit Schematic for Pulsing Lights
Figure 6  LED Light Fading Circuit Schematic
Figure 7  Final Metronome Presented at Engineering Days
Figure 8  Final Metronome Close-up at Engineering Days
Figure 9  Metronome and Poster Being Presented at Engineering Days
Figure 10 Price Breakdown of Components Used in the Final Metronome Before Shipping
Abstract

A metronome is any device that produces a regulated audible and/or visual pulse that is usually used to establish a steady beat or tempo for the performance of musical compositions. Originally invented by Dietrich Nikolaus Winkel in 1812 this device has progressively been evolving until the now widely used electronic version that is more sophisticated.

We decided to expand upon this idea to create a new metronome with many specific functions all in the intent to aid in not only keeping time for the musician but to also teach the musician how to become a better player. Using mode selections the user is able to switch between a standard metronome, a hit counter, a rhythm playback, a tap tempo, and light show all controlled by the user.

Cypress semiconductor has produced a new line of microcontrollers called the PSoC (Programmable System on Chip) which is essentially a family of mixed-signal arrays, featuring a microcontroller and integrated analog and digital peripherals. The PSoC is a complete system contained in a single chip and was perfect for the design we had in mind. It is with this chip that we were able to create our version of the metronome called the IMP^2 that will hopefully be a marketable device that everyone can use.
I. Introduction

The PSoC is a new chip created by Cypress Semiconductor and it has become a brand new senior design project started by George Collins and given to us to implement whatever design we feel was fit to explore all the functionalities of the chip. By exploring the possibilities of the chip our group consisting of Brian Campbell, Michael Kabala, Justin Mirarck, and Robert Utrup have decided to create an electronic metronome that would show the capabilities of the chip.

As part of our project requirements, we learned the functionalities of the PSoC to fully understand what can be done with the chip and how hard we can push it before we began designing the metronome. Then, once we understood our goals we created the metronome with integrated hardware to change between the 5 different modes the device would do.

The concept for our metronome is to create a device that would not only have a generic metronome but also have a tap tempo, hit counter, rhythm playback, and a light show all of which will be user controlled explained in full detail later within the report.

Through the countless hours we have spent on the design measuring, researching, and experimenting with our results we have come across many obstacles that we overcame to create a working prototype of the design. The task was arduous but in the end the rewards that sprung about from it made it worth all the while.
II. Summary of Previous work

To summarize our previous semester’s work we researched the capabilities of the PSoC chip, came up with a rough design of our metronome, experimented with the PSoC, and made a rudimentary metronome with a single function.

Most of our time last semester was spent on the researching the full functionality of the PSoC. During the research portion, we learned about the different capabilities that the PSoC has, such as: Digital to Analog converters (DAC), clocks, pulse width modulators (PWM), and memory. This was done by working through various example programs already contained within PSoC Designer. A good understanding of how everything worked together was achieved, by going through the example programs. Later, we obtained the C compiler license and began reviewing examples in C as well.

While researching the PSoC, our group struggled with developing a solid idea for our project that we could all agree upon. Finally, Justin devised the idea of creating a metronome with various features that currently are not available in the market. This idea was well received, as we would be able to utilize each part of the PSoC.

During winter break Mike synthesized a basic metronome using a simple 555 timer. This simple example didn’t have the ability to change tempo, but it was useful to test the various hardware circuits that would later be built.

III. Project management

The design of the project was done in two ways. The first way was dividing up the tasks to the individual team. The second way was working as a team to bring the
individual tasks to a completed design. As far as this semester the tasks that have been
assigned are as follows:

Brian Campbell – For the second semester of senior design Brian Campbell
continued in his role as the group leader of the project. Brian was in contact with our
senior advisor Cameron Moore. He was able to contribute to the team by helping Robert
with some of the programming and working on debugging our signal processing problem.
He was also able to lend advice to each team member and facilitate group meeting to
make sure the project got completed in time for E-days. Brian also helped solidify the
final look and feel of the metronome. There were many different ideas about the final
product and Brian and Mike were able to make the final design choices with input from
the team members. Brian was also in charge of purchasing almost all of the materials for
manufacturing the completed metronome. He was able to find good deals on different
parts as well as being in contact with Robert, Justin, and Mike regarding which parts each
person would need.

Michael Kabala- The second semester Mike contributed to our project by heading
the design on the digital circuits that would control the mode switching buttons as well as
the light effects that added some of flash and visual appeal that our project had. Mike
took up the design work on our logic circuits and excelled at building up the complex
circuits. Mike was also a valuable asset in the manufacturing of the final product and
took charge of some of the enclosure design. Mike also contributed greatly to our project
with his graphic design on our posters and graphics that we used for both E-days and
exploration days. Mike was able to also work with Robert on filtering the drum signal
that we used to detect hits. Mike worked on building different filters and op-amp circuits
that we intended to use to filter the signal, but none of the designs worked well enough. This eventually led us to move toward coding and software for the sampling of the drum signal.

Robert Utrup- During the second semester, Robert Utrup led the design with the PSoC chip. Robert took up the role of learning the intricacies of programming the PSoC chip with both the PSoC designer and with C programming. Robert was able to put in many hours of examining example C code and learn how to modify and use this code to create the different aspects of our metronome. Robert first programmed the metronome mode and then found other aspects of a metronome that could be programmed. He learned how use different user modules that controlled clocks, delays, analog to digital controllers, and pulse width modulators. Robert was instrumental in leading our programming and taking ideas from Justin’s musical background about different features that our metronome could contain.

Justin Mirarck- Justin Mirarck contributed the second semester by helping Mike with the filters and op-amps that Mike was building. Justin also is given great credit with coming up with the idea of the metronome for our project. Justin is a drummer in a successful band located in northern Colorado and was able to work with Robert in designing the different features of our metronome. Justin also helped out greatly in both our E-days and High School Exploration days by bringing in his drum set and interacting with judges and students about the need for our metronome and the new features that our metronome have. Justin also was in charge of the soldering and contributed greatly to the manufacturing of the final product. Justin had experience in soldering from work and
was best suited for this task. Justin also built the project website and updated it through
the semester.

IV. What is the PSoC

The PSoC, which stands for Programmable System on Chip, is a family of mixed-
signal arrays made by Cypress Semiconductor, featuring a microcontroller and integrated
analog and digital peripherals. The PSoC is a complete system contained in a single chip.
With some microcontrollers one must include elements such as D/A converters but with
PSoC’s everything is contained and ready to be used. It is important to think about the
total system in two distinct parts before looking more in depth at the actual PSoC chip
itself. The first part is the computer with the PSoC Designer program used to write the
code and transfer it to the chip. The second part of the system is the board with the PSoC
chip on it. Once programmed, the PSoC can perform a multitude of tasks including,
taking measurements, turning and output on, and also reporting back to the computer.
Now that we have taken a brief look at the total system, we can examine that actual PSoC
chip and its attributes and capabilities.

The PSoC chip is a device that includes user configurable blocks of analog circuit
and digital circuit blocks. The chip also includes programmable interconnection between
these parts of the chip. We will look at the configurable blocks of analog and digital
circuits in more detail when considering user modules. The PSoC chip also contains a
CPU, flash programmable memory, and configurable IO which can be used to control
sensors and output.
Specifically the PSoC chip that we are using is the CYU6983. This chip comes in a 48 QFN (Quad Flat Pack No Leads) package. This is a small footprint for a chip. This smaller packaging is difficult to solder into a circuit board by hand and usually needs to be done by a company that has soldering abilities. Looking at this Cypress chip from a top level architect standpoint there are a few main components which require attention. The chip contains a PSoC core, digital system, analog system, and system resources.

The PSoC core contains a large instruction set. It also has SRAM for storage and an interrupt controller. The interrupt controller can activate sleep and watch dog timers as well as execute new programs. The core also contains multiple clocks including a phase locked loop, internal main oscillator, internal low speed oscillator and external crystal oscillator. These clocks are valuable in implementing various timing applications. The core can clock up to 24 MHz. The core is called the M8C and it is a four MIPS 8-bit Harvard architecture microprocessor. There are programmable interconnections between the pin-outs and the main CPU which allow for many different output applications as well as flexibility.

The digital system of our Cypress PSoC chip is made up of digital rows in block array and utilizes the Global, Array, and Row Digital Interconnects. Our chip has the ability to route any block of digital systems to any output pin through global buses which allows for multiplexing and logic operations. This allows for flexibility in control of various outputs and sensors.

The analog system is composed of analog columns in block array. The system also includes analog references, analog input multiplexing, and analog drivers. On our PSoC chips we are constrained to a maximum of four analog columns with up to twelve
analog blocks. Looking at these analog blocks in more detail, one will find that each block is composed of an op-amp circuit which allows for flexibility in creating analog signal manipulation. Looking at the column shows that each contains one continuous time block, Type B, one switched capacitor block, Type C, and one switched capacitor block, Type D.

Finally, we will consider the last component of our chip the system resources. The chip contains up to four MAC’s or multiplying accumulates which allow for rapid 8-bit multipliers. Also, the chip contain up to two decimators which can be using in digital signal processing applications. There are slave and master capabilities as well as 1.3V of power which can be used for various PSoC subsystems. There is also a switch mode pump that generates normal operating voltages off of batteries. This is used in the stand alone part of the total PSoC system as this system is completely severed from any base station. The chips contain enhance analog multiplexers which allow for every IO pine to connect to a common internal analog multiplexing bus. The chip supports a full-speed (12 Mb/s) USB device. There are also resets in the system.

Some final high level specifications of our chip include voltage, current and data transfer rate. The operating voltage of the chip we used for our project is 2.7 V to 3.6 V, and the standby current is less than 1 µA. The maximum rate of data transfer can be as high as 62.5 kbits/sec. The ideal operating temperature ranges from 0 to 70 degrees Celsius.

“PSoC Designer(TM) helps users harness the power and flexibility of the PSoC device by providing ‘point and click’ system design capability. It includes pre-configured, characterized peripheral functions in the form of User Modules and extensive
user assistance in the form of Help dialog boxes, pull-down menus, and other GUI aids.”

As can be noted from this quote from Cypress, PSoC Designer is the program we are using to program the chipset. It is a quite powerful program while simultaneously being very user-friendly and easy to learn. It is split into 3 important sections: Device Editor (where all the user modules are placed and configured), Application Editor (where the main program is created in either C or assembly), and Debugger (which can be used in the design process to discover errors). I’ll discuss them in brief.

As mentioned previously in this report, the device editor is where all the user modules are configured. The user has many options when selecting the projects modules. Most of them are connected to the pins of the board, either as in input, output, or both. A few examples of modules that can be connected to the ports are: Pulse Width Modulators, D-to-A and A-to-D converters, and the LCD module. However, there are also other modules such as the counter that can be strictly used within the program and have no connection to the outside world if desired.

Once a module is selected from the pull-down menu, it generally must be placed somewhere in the grid provided. Once placed, the user can then make connections to other ports. This is also the time when the parameters are chosen, such as clock speed, and in the PWM’s case: period, pulse width, compare type, etc. It is also of use to state that each module has a great help section devoted to it. It discusses how to configure it, both assembler and C code for controlling it, and there is also a little example code given to have it perform a task. Now that all this is complete, it is time to move onto the application editor.
Figure 1 PSoC Designer: Device Editor

The Application Editor is where all the coding is done, where the true brain of the system is. Before starting this project, the user decided on using cypress’s assembler, or using the C compiler created by ImageCraft. Regardless of the choice, the functionality is the same aside from different code being needed to control the modules. This area can be quite complex, but cypress has done a good job of making it seem simple. All you have to do is put in code for the main program, all the other coding such as library headers and the boot file are already taken care of for you. However, if you found it necessary to alter these system files, PSoC Designer helps with this. The most interesting
part of the application editor is how it is responsible for controlling the user modules while the PSoC is running. It has the power to dynamically reconfigure them. For example, it can not only turn individual modules on and off at any time, but it can also change the power they are running at as well as completely get rid of one while interchanging it with another module. This gives the user the ability to completely control their system; one benefit of this over other hardware is the possibility to go into a low power mode when the conditions are found to promote it. Now if everything goes according to plan, your experience with PSoC Designer ends here. The programmer is opened up and the board receives the instructions you just built for it. However, if there is a problem then the debugger comes into action.
We have not yet found it necessary to use the debugger. All our problems with the code were solved with a little old fashioned trial and error. However, the debugger is a powerful tool none the less. It has all the options one would expect from a debugger, such as break points for pinpointing the error in the code. As a side note, the debugger has the ability to interact with Cypress’s ICE-CUBE, which stands for Integrated Circuit Emulator. It allows for the code to be run through step by step and to see and change the data each time.

In our project we have been able to learn about the possibilities of the PSoC chip by examining projects and designs and other people have been able to carry out using the Cypress PSoC chip. Cypress is able to promote their chips by showing the various uses that they have. The Cypress website has application notes that individuals have created and submitted to Cypress. There are many applications submitted on this website and there is incentive for submitting creative and useful application notes.

Cypress offers the incentive to individuals to develop projects and designs with their family of PSoC chips. Cypress will pay $250 for a quality application note that is a minimum of two pages in length. This application note includes the overview of the project as well as pictures and schematics of the final working design. Individuals also include the working code for their projects. Essentially one could take an application note and build a design strictly from the application note. Our senior design team has the possibility of submitting an application note at the very end of the year for consideration by Cypress.

One application that has been implemented using the PSoC chips from Cypress is an OLED Demo Board. This mixed signal array can run the display and receive images
through a USB connection to a computer. This can be accomplished with the Cypress chip that our project is using. Another application that I found was a magnetic compass with tilt compensation. I also found a project that utilized wireless mixed signal arrays to gain keyless entry into a car. This design implements a spread band signal to create a secure keyless entry. This system implements the Cypress chips and is much more difficult to defeat by criminals.

V. The Metronome

After learning about the PSoC and what it can do we needed to create a project from this that would adequately perform the majority of the chips functions. For our senior design we wanted to come up with a product that could demonstrate many capabilities as well as be a marketable product that would benefit many people. Our idea came to us after many days of thinking. Since everyone in the team has some musical background we decided to create a programmable metronome seeing as how each one of us has used one and wished there were other options within the metronome that seems like it should be there. With many more days of deliberation we were constantly thinking of ideas that the metronome should have that would benefit the musician in more ways than one. The idea we came up with were to have several modes that all change to a different function that the user would be able to use. The modes we came up with were a standard metronome, a tap tempo, a hit counter, a light show, and a rhythm playback all of which will be explained in greater detail below. There were many challenges we had to overcome, such as power consumption, memory size, and dynamically changing the pulse width modulator’s duty cycle to achieve the different tempos.
VI. Software and the Modes

The PSoC is the driving force of our metronome for the basic fact that it does all the computations and algorithms necessary to execute each of the different modes. As mentioned the Metronome has five different modes with expansion slots to implement many more.

The first and most obvious mode for a metronome is of course the metronome mode. It keeps tempo for musicians from the typical range of 60 BPM to 250BPM. This is achieved by taking the voltage across a variable resistor and sending it into an A/D converter. We used this as a knob the user could turn to change the tempo. Once this position is recorded, it is converted into a BPM which can then be displayed on the LCD screen and outputted by flashing an LED and a little speaker. The actual tempo is created with a pulse width modulator, or PWM from now on. The PWM keeps a standard pulse width, so each beep sounds uniform. The change comes from either shortening or lengthening the period. A shorter period corresponds to a faster tempo, and a longer one goes to a slower tempo. A lot of calculations were performed to divide clock cycles properly to get the metronome accurate. Once it was developed, it was tested on sample tempos and it was found it was accurate to within 1 beat at 120 BPM in a 60 second range. We found this to be quite acceptable, a negligible difference.

The second mode, and probably the most important for making this design differ from thousands of others, was called Tap Tempo. Tap Tempo, as the name implies, allows the user to tap on a drum pad 4 times to create a tempo. This is done instead of manually adjusting the metronome. This mode was the most crucial, and subsequently, the most difficult to create. As a top-end explanation, the PSoC takes in a wave from the
drum pad, and records the time each time it is hit. After 4 hits it averages the time between each hit and creates a tempo which is then played out with the PWM, just like the metronome mode. However, this explanation doesn’t cover all the little intricacies that troubled us so much with it.

The drum pad was piezo resistive, and it would create a sinusoidal wave within a decaying envelope each time it is struck. Furthermore, when connected to the A/D converter of the PSoC, a capacitance was added to the circuit. Therefore, now when it was hit, a constant voltage was created, because no resistance was available to bleed this voltage down. This was obviously not useable, so a resistor was calculated that would be introduced into this circuit to create a time constant RC. Now, when the drum was hit it would spike up fast and then moderately drop till zero voltage was reached. We had the option of creating an interrupt each time it was hit, or poll every 10mS for hit data. We chose polling due to the noise creating false interrupts and the realization that we could sample fast enough to simulate a continuous function over our frequency. Originally, we thought this would be all the solution needed, a threshold could be created, and every time the signal went above it, a new hit would be realized. Unfortunately again, this wasn’t the case. This solution worked for slow inputs, but if a user were to rapidly hit on the drum, a wave form such as the one below was created.
When this occurs, the output never drops below the threshold, and all hits afterwards are ignored. It was then obvious a more complex algorithm had to be created to solve this issue. We started by looking at the waveform. It was obvious each time the drum was hit, so we knew there was a way to make the PSoC realize when it happened. The answer finally came by playing out several scenarios on paper. An example waveform would be drawn, and sample points would be drawn on it. We would then follow each point, and figure out what the computer should think of it. Should it be expecting a new hit? Should it be ignoring the data? Is this a new hit? Etc…

Here is the algorithm: For simplicity, let’s give the noise an A/D value of 40. We’ll now say that anything above it isn’t noise; it is a change in the waveform. We chose 50+ to show this. As soon as the wave increases 50, it is a new hit, record the time. Next, ignore all of the subsequent increases of 50+ until it drops at least 50. Once this
occurs, we know that the first hit has finished and now is declining. Next, we can look for a new hit, which to be registered must be 50+ from the previous sample value. With this algorithm, the drum can be hit as fast as a human can hit it and the sampling and signal processing will be fast enough to handle it.

To wrap up this mode, the computer takes in the 4 hits, and the precise time (within 10mS) that they occurred. They are all added and divided by 4 for an average and then further worked upon to transfer it into a BPM ultimately useable by the end user. This mode gives power to the user who has a beat in his/her head but doesn’t know the BPM value. It can be used to test the accuracy of the musician, or simple used to start the tempo, similar to hitting a pair of drum sticks together 4 times.

The third mode was created with the knowledge that we would be presenting this project several times. It was invented to get an audience’s attention, and let them interact with IMP^2. This mode is called Hit Counter. Just as the previous two modes, we picked its name to be as obvious as possible. The user has 10 seconds to hit the drum pad as many times as possible. Once the 10 seconds is complete the total hits and its corresponding BPM is scrolled across the screen.

This mode functions based on many of the tools created for the previous modes. The concepts in Tap Tempo are extrapolated upon and tweaked to create the desired results. The same algorithm is used to see when hits occur and an integer value is incremented each time. Also, a counter is incremented each time the polling occurs, so that 10 seconds is precisely measured.

It is worth mentioning that this mode has limited success with registering double hits, triple hits, and rolls. All these are similar in that the user hits the drum but keeps the
grip loose, so as to hit the drum more than once in rapid succession. The sampling is done quickly enough, but when this technique is used the hits are often times so soft that they don’t register above the noise level. A solution that never got implemented would be filtering. A low pass filter could be used to reduce noise and allow softer hits to be received by changing the noise level to <10. Possibly even some Digital Signal Processing within the PsoC could have provided better results.

The second to last mode is called Rhythm Playback. This features the capability of playing any sort of rhythm on the drum for 10 seconds. It is then processed and played back to the user with the LED and speaker. All the mechanisms that allow this have been discussed already, so strictly the new parts will be explained.

Interrupts are continuously happening in this mode. During the 10 seconds, interrupts occur to look for hits. When a hit occurs, the time it happens is put into an array. The structure looks like this: Array_Name[1] = t1   Array_Name[2] = t2   etc….. So now it is convenient to pull data from it, hit one is [1], 10 is [10] etc…

The playback is where it gets a bit trickier. An integer value is put into the array and it starts at 1. The time is set back to zero and the interrupts start occurring again. As soon as the time of the interrupt is the same as the recorded value, an output on the PWM is made. Next, the variable in the array is incremented to the next hit, and the system waits for when this new time occurs. This process is repeated until the 10 seconds is fully synthesized and played back.

The final mode created was called Drum Lights. Possibly the most worthless of all modes, the sole purpose for it was to create a bit of a light show to be pleasing to the judge’s eyes. Every time the drum was hit, a pulse would be sent to a logic circuit, and
the top-mounted LED’s would advance one space. This had the effect of making a swirling light around the box when the drum was hit very quickly. The same logic behind the other modes involving the drum was used to distinguish when a hit occurred, and a brief pulse from the PWM was created.

Several other ideas for modes were brought up, and although plausible, they weren’t implemented. One idea was to allow for the volume on the Rhythm Playback to change based of the intensity of the hit. A basic working system was devised, but due to problems with the hits being recorded at various voltages (due to speed of hits) this idea was ultimately scrapped do to poor performance. It was possible to implement, but not without revising the entire project.

The second idea we all liked was a mode called Simon Says. There would be a few features such as the challenge of hearing a rhythm played to you, and doing your best to match it. Once you finish the computer would compare the two sequences and give a percent accuracy to the user. This mode posed a whole lot of design issues, all of which were feasible but horribly tedious. This idea was made near the end of the project, and it simply didn’t have enough time to be fully created.
VII. Code Statistics

2 ADC’s running at 1.5MHz
2 PGA’s
16bit and 8bit PWM’s both at 750Hz
1 8bit Timer running at 750MHz and causing interrupts every 10mS
2 analog inputs
5 inputs for mode change

Pins:

Port 0:
pin 5 >> LED/PWM (OUTPUT)
pin 3 >> VR (INPUT)
pin 4 >> Drum >> Parallel 5.5Mohm (INPUT)

Port 1: Mode Button Inputs
pin 7 >> Metranome
pin 6 >> taptempo
pin 5 >> hit counter
pin 4 >> Playback
pin 3 >> Drum Lights

Port 2: LCD Outputs

Language used: Both C and Assembly
Number of variables: 24
Number of while loops: 20
Number of if loops: 35
Lines of code: 672
Number of pages: 13
VIII. Hardware

The hardware used in the metronome is essentially there to change the modes of the device. Since we have 5 modes with room for expansion we needed a system that would be able to tell the PSoC when a button is pressed to change the mode. This was done by having a set threshold voltage in the 5 inputs so that when a button is pressed it will correspond to a high value as seen by the PSoC. The corresponding input port where the buttons are hooked up will be able to give an address for the program to jump to whatever mode it needs to depending on what port is receiving a high value from the button system.

The system itself consists of several D flip-flops, NAND gates, inverters, and AND gates. What would happen is when a button is pressed it would send a pulse to the D flip-flop initializing a 5 volt output to the PSoC input port. What makes this system unique is that when the button is pressed it would also send a pulse to the other 4 D flip-flops where through a series of NAND gates and AND gates would initialize the reset function on the D flip-flop turning them off so the only port that should be on is the one where you pressed the button. This also means that on the fly you can push another button and it would reset all the other D flip-flops which in turn would reset the input ports on the PSoC registering a 00000 value for a split second and then initialize a high value to the proper address corresponding to the button that has just been pressed.

We experimented with several other types of buttons systems such as a single button that would be able to change to a different mode with the press of a button however this had several flaws in it. The first flaw was dealing with the issue of bouncing. When the single button is pressed it essentially bounces several times in
millisecond delays and would change the mode 2-3 times skipping the one you really want to get to. We tried filtering out the bounce using a de-bouncing circuit however it was not clean enough to solve the problem. The other issue was user ability. When the button is pressed it would change to the next mode that you programmed in a set mode sequence. However if the metronome had 10 modes, the user would have to cycle through all 10 modes to finally reach the one he/she wants instead of just pressing a button corresponding to the certain mode they want. Therefore, you would also have to add another button to go backwards in the sequence just incase you went passed the mode you wanted. By doing this you would then already have to add another button in turn making the circuit larger. The button system that is in place now seems to work the best for its reliability and no bouncing upon pressing a button.

On top of this we experimented with several buttons as well. The first buttons we had were single pull double throw button but with this one you had to press the button twice to deliver the pulse to the D flip-flops. Then finally we went with the single pull double throw momentary switch that has a spring inside to reset the button to its original state after it has been pressed delivering the sharp pulse into the D flip-flop. In any case there was much experimenting to find the exact buttons and logic gates to accomplish this system but in the end worked out perfectly.
Besides the buttons system there are other hardware components involved in the metronome that go along the lines of the overall visual appeal. We figured that since the metronome right now is only giving audible sound we want to incorporate a visual aspect of it just incase the user cannot hear the tempo through the music. This circuit is composed of a decade counter and a string of 10 LED’s that would light up in sequence every time a beat occurred. The decade counter itself is taking in the same pulse that is
driving the piezoelectric buzzer and using it to switch to a different LED in the sequence and light it up.

Figure 5: Visual Circuit Schematic for Pulsing Lights

The final piece of hardware that we created was also used for visual appeal and this was the LED light faders. These circuits cycle in different frequencies regulated by resistor values to light up several arrays of different colored LED’s. We used an LM324 amplifier which is similar to the 741 operational amplifiers; however it is ideal for these LED light faders. With a combination of NPN and PNP transistors to switch between different colored LED’s as well as several combinations of capacitors and resistors, we were able to control the rate of change through the charge and discharge of the capacitors through the transistors to get several hundred different colors as they faded in and out of each other. The purpose of these circuits is to illuminate the metronome at night but to
also give the product appeal to a younger crowd such as small children to hopefully help them learn how to play an instrument.

![LED Light Fading Circuit Schematic](image)

**Figure 6: LED Light Fading Circuit Schematic**

These hardware circuits in combination to the PSoC they are attached to created the metronome that has 4 different functions with the capability of expansion.

**IX. Final Design**

When we were finally done implementing the software to the modes that we wanted and finished constructing the hardware the final procedure was to assemble the metronome into a prototype enclosure. The optimal box dimension that we needed to enclose everything was about 3.38” x 8.5” x 12.38”. To do this we could either create a
box from Plexiglas ourselves or buy a pre made enclosure. Due to time management we ended up buying an enclosure made up of acrylic plastic.

The very first thing we had to do was drill holes for the buttons, LED’s and LCD screen to be mounted. This was done using a standard drill spinning at a high rpm so that no cracking would occur. Once this has been done we then continued to place all the LED’s and breadboards within the box in predetermined locations to minimize wire interference. To keep everything together we decided to solder all the LED’s to long solid wires that we ran back into the protoboard wound up with heat shrink tubing to prevent the tangling of these wires. Originally we were going to have the electric drum pad sitting on top of the box however this would greatly reduce the area of which to work in so therefore decided against it and left the pad on a stand. By doing this the drum pad was properly supported and gave more accurate outputs as well.

Everything seemed to fit exactly where we wanted it however the only major problem we had was keeping the wires of the LCD screen in tact as the box was being closed. There were so many wires that they would literally push up against the wiring of the LCD screen and disconnect it. This was a major problem since we didn’t want to solder each individual lead but instead us electrical tape to keep the wires together. By removing some wires stationed below the LCD wire lead we were able to free up extra room for the LCD screen to sit comfortably without disconnecting any leads that would prevent the system to work.

The original design was going to have the IMP^2 logo etched in by sandblasting however due to time constraints a valid substitute of sandpaper hand etching worked perfectly. The overall box turned out better than we expected and was a true site to
witness during the Engineering Days presentations where we received many compliments.

Figure 7: Final Metronome Presented at Engineering Days

Figure 8: Final Metronome Close Up at Engineering Days
X. Design Challenges

A great challenge we faced the entire year was receiving our wireless development board from Cypress. The company had told us that we were going to receive this board early this semester but we kept running into red tape that slowed down the delivery of this item. It was frustrating to be totally halted when we had wireless chips and antennas ready for implementation. We were not able to purchase the development kit ourselves since it cost over $700 dollars. One note is that the item finally came in last week. This challenge taught us about real world situations involving receiving parts to complete a task. We were able to overcome this obstacle by making the best wired metronome that...
we were able. Modifying a design is viable option and compromising is not being defeated in accomplishing our goals.

Another, more technical challenge, was implementing a method to realize drum hits. The final solution was discussed earlier in this paper, but the road to solving it was difficult. We originally started by devising hardware solutions, but they all fell short. While not perfect, our final solution in the form of a software algorithm performed up to the specs we needed.

Finally, one of the last challenges we saw was when the mode switching abilities were being created. The system takes in voltages from the hardware, and the proper pin is set to high to specify that mode. This worked perfect, but when combined with the drum pad, it had serious issues. When the drum pad was hit, interference would occur on the other pins of the PSoC, thereby giving false values to the program and causing random mode changes. This problem was ultimately fixed with more of a band aid then complete fix. Everything was kept the same, but the program would only look for mode changes at specific points. By doing this, the mode cannot be switched while the drum pad is being hit. This was fine for display purposes, but the ease of use was degraded by taking longer to switch modes.

XI. Budget

In the senior design projects, each team member is allotted $50 dollars/semester. So our team is able to utilize $400 dollars total for the entire year. Since many parts that have to deal with engineering don’t come cheap such as microprocessors, programming boards, software, and passive and active circuit elements we have to make sure we buy
everything we need at a reasonable price. To do this we had to research many of the parts that we believe we will be using in order to find the best deal.

Many of the parts we bought were in the final stages of our design such as the LED’s, enclosures, and circuit elements. At first we got in contact with Cypress semiconductors and we received a PSoC evaluation board as a donation from Cypress and were able to use this to start designing our project. We also received a donation of a PSoC C compiler license which allowed us to start writing C code and examining C code examples. Cypress also issued us 3 extra PSoC chips at no charge just in case we burnt the original one out or needed more to accomplish our tasks. We were hoping to receive a Wireless Development kit from them however Brian Campbell contacted Cypress and received free donations of wireless chips including antennas but we were never able to implement them because we didn’t receive the development kit so wireless was not an option for us anymore. As far as other components go we were able to obtain everything we needed at distributor stores around town such as Eagle High-Tech and Mountain State Electronics here in Fort Collins. The parts we were able to purchase from them included the logic gates used for the button system, amplifiers, wires, buttons, and solder. The majority of our budget went for the LED lights and wires which were purchased towards the end of the semester. We ended up using over 30 LED’s in the metronome and over 200 feet of wiring to accomplish our task. The get these parts we went to the website www.oznium.com who specializes in LED lighting and other visual peripherals. Shown below is a list of the components all purchased and used in our final design, where you can see we were still under the budget we were given.
<table>
<thead>
<tr>
<th>Product</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSoc development board</td>
<td>Donation</td>
</tr>
<tr>
<td>Resistors</td>
<td>Free from school</td>
</tr>
<tr>
<td>Electric Drum Pad</td>
<td>Free from team member</td>
</tr>
<tr>
<td>Buttons (SPDT) $3.99*6</td>
<td>$23.94</td>
</tr>
<tr>
<td>LED’s from Oznium.com</td>
<td>$157.48</td>
</tr>
<tr>
<td>Wires 4 x $3.87</td>
<td>$15.48</td>
</tr>
<tr>
<td>Breadboard</td>
<td>$13.99</td>
</tr>
<tr>
<td>Solder</td>
<td>$13.95</td>
</tr>
<tr>
<td>Potentiometer</td>
<td>$3.50</td>
</tr>
<tr>
<td>Turning knob</td>
<td>$2.59</td>
</tr>
<tr>
<td>NAND Gates 6 x $.66</td>
<td>$3.96</td>
</tr>
<tr>
<td>AND Gates 6 x $.45</td>
<td>$2.70</td>
</tr>
<tr>
<td>Inverters 4 x $.55</td>
<td>$2.20</td>
</tr>
<tr>
<td>Decade Counter</td>
<td>$1.15</td>
</tr>
<tr>
<td>Amplifier 5 x $1.25</td>
<td>$6.25</td>
</tr>
<tr>
<td>Transistors 8 pk</td>
<td>$2.15</td>
</tr>
<tr>
<td>D flip-flop 6 x $.75</td>
<td>$4.50</td>
</tr>
<tr>
<td>Enclosure</td>
<td>$17.95</td>
</tr>
<tr>
<td>USB Cable</td>
<td>$26.66</td>
</tr>
<tr>
<td>8 pk cushioning feet</td>
<td>$2.19</td>
</tr>
<tr>
<td>Item</td>
<td>Price</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>Mounting tape</td>
<td>$1.99</td>
</tr>
<tr>
<td>DC chime</td>
<td>$9.99</td>
</tr>
<tr>
<td>¼ inch shrink wrap</td>
<td>$2.95</td>
</tr>
<tr>
<td>3/16 inch shrink wrap</td>
<td>$2.35</td>
</tr>
<tr>
<td>Black tape</td>
<td>$1.10</td>
</tr>
<tr>
<td>3/32 inch shrink warp</td>
<td>$1.70</td>
</tr>
<tr>
<td>Speaker</td>
<td>$3.20</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>$324.00</strong></td>
</tr>
<tr>
<td><strong>Budget Left</strong></td>
<td><strong>$76.00</strong></td>
</tr>
</tbody>
</table>

**Figure 10: Price Breakdown of Components Used in the Final Metronome Before Shipping**

**XII. Manufacturability and Marketability**

In its current state our project is still in the prototype stage. For our prototype we wanted to build a box to enclose our protoboards and circuitry. Our original thought was to use lexan plastic so the inner workings of our metronome could be visible from the outside. As we researched the lexan material we found that it is fairly expensive. To save money we then looked into and purchased a pre-made case for our project. The case we bought was an acrylic box with the dimensions 3.38" x 8.5" x 12.38", it had to be big enough to fit the two protoboards, the PSoC evaluation board, and all the wiring. To wire the all the components together it took over 100 feet of wire. Obviously, this would not be a very practical design to manufacture. Given more time we could have taken our project to the next level by reducing the size of the circuitry to create a cost effective and
overall smaller product. To do this we would need to use technologies such as: Programmable Logic Array (PLA), a Hardware Description Language (HDL), or a PIC microcontroller. With all of the circuitry in a few chips they can be soldered on a Printed Circuit Board (PCB) that uses layers of copper for the wiring from chip to chip. Doing so will dramatically decrease the overall size of the circuitry, thus allowing for a smaller enclosure to further decrease the cost.

For the tap tempo, hit counter, and rhythm playback modes of our metronome, an electronic drum pad is needed as an input. Trying to build an electronic drum pad would no easy task. Luckily for us, there are several companies that already manufacture them. Developing a partnership with such a company would be very beneficial.

Our original intention was to market this product to musicians, but our minds were changed after E-days. During E-days a group of young kids approximately 8-13 years old were swarming our booth for the majority of their time at the event. The kids were mostly interested in the hit counter mode; it turned into a competition to see who could get the highest. With little change to the circuitry and enclosure the metronome could be tailored to fit the needs of a musician, or the requests of the youth.

We experimented in several occasions what people thought about our project in a large group demonstrations. One instance was when our group was asked to participate in Engineering Exploration Days. On this day, high school students who are considering engineering at Colorado State University come to campus to learn about our engineering program. A select few Senior Design groups from each department were asked to demonstrate their projects for the prospective students. We accepted the offer because we thought it would help us out in several ways. It proved to be good practice for E-days.
We were able to work on explaining and demonstrating our project for an audience. We were also able to gain feedback from several different perspectives. Nearly all of the feedback we received was good, so we concluded that our project was headed in the right direction.

XIII. Conclusion

In conclusion, our team has leaned many valuable skills and lesson from this senior design project. We were able to complete our goals of making a metronome as well as utilizing the PSoC’s capabilities. Our group overcame different obstacle such as availability of parts, size of design, and making time to work on the project amidst our busy schedules. Robert was able to learn from our project about relationships with parts vendors and how companies can let one down and crucial times. Justin learned how different expectations can change in the course of designing the project. We were not able to make the project as small as we wanted to but either way it was quite an accomplishment. Michael learned about using trial and error in building circuits to complete signal processing. He also learned about the challenges of building logic circuits and the satisfaction that comes from success when everything work the way it should. Brian learned about directing a group and trying to assist everyone in their roles. He also learned about talking with companies and ordering parts for the project. Overall our team feels that our project was a success. We started with and idea, put in considerable time, effort, and ingenuity, and in the end because of this hard work we were able to design and completed a working product known as the IMP^2.
XIV. Works Cited


"Design Resources." Cypress Semiconductors. 13 Feb. 2007
&parentname=CommunityPage&parentid=1&in_hi_userid=2&control=SetComm
unity&CommunityID=285&PageID=0>.

