

DEMONSTRATING CAPABILITIES OF ISAAC (INSTRUMENT SHARED ARTIFACT FOR COMPUTING)

Chester N.V. Lim
Jet Propulsion Laboratory
Department of Electrical Engineering
University of Hawai'i at Mānoa
Honolulu, HI 96822

ABSTRACT

ISAAC is a research and technology development project whose objective is to provide a FPGA-based computing and control platform that will control a variety of onboard instruments ranging from imagers, radar and radiometers. There are six sub-parts which makes up ISAAC (iBench, iCore, iPackage, iBus, iBoard, iTool) each performing separate tasks. The objectives for this summer are to assist the engineers who are working with the iCore and iPackage sub-parts. iCore is a sub-part that holds the library of RTL (Register-Transfer-Level) codes and is reconfigurable depending on the specific instrument. The task for iCore is to help with the verification, validation and resource usage estimations. iPackage includes a branch called iPanel and its purpose is a high level interface in which the user will be able to gather telemetry from the instruments. The task for iPanel is to create the GUI (Graphical User Interfaces) in Python that will display telemetry from the instruments and be able to have a command console for the user to send instructions to the instruments. Another task for iPackage is to put together a demo for the FPGA computer platform, which includes being able to gather telemetry from an I2C thermal sensor and UART motor.

INTRODUCTION

This paper will discuss about the work that as been done over the ten weeks span of this internship. The objectives for this internship was to assist Dr. Yutao He and his team in developing a highly capable, reusable, and modular FPGA-based computing and control platform for instrument digital electronics. The FPGA based technology called ISAAC will be used for signal processing, telemetry collection and motor control. The tasks for this internship are to assist Jason Zheng and Kayla Nguyen with the iCore sub-part and to assist William Zheng with the iPackage sub-part.

Two types of chips may be used for an implementation on an application and they are hard-wired chips and programmable chips. Hard-wired chips, like an ASIC (Application Specific Integrated Circuit), are designed for a specific application and are not ideal for modularity and not reconfigurable. Programmable chips like an FPGA (Field Programmable Gate Arrays) on the other hand allows more room for modularity and are reconfigurable.

The advantage of using an FPGA based technology is that it is possible to assemble the same FPGA devices to be used in different applications thus reducing the non-recurring engineering costs. In addition, unlike ASIC's, if there is a mistake with the logic, FPGA's have the ability to be reprogrammed. An FPGA device may also act as a redundant system for a random failure in a certain logic device. The redundant FPGA device may be reconfigured to take over the failed device.

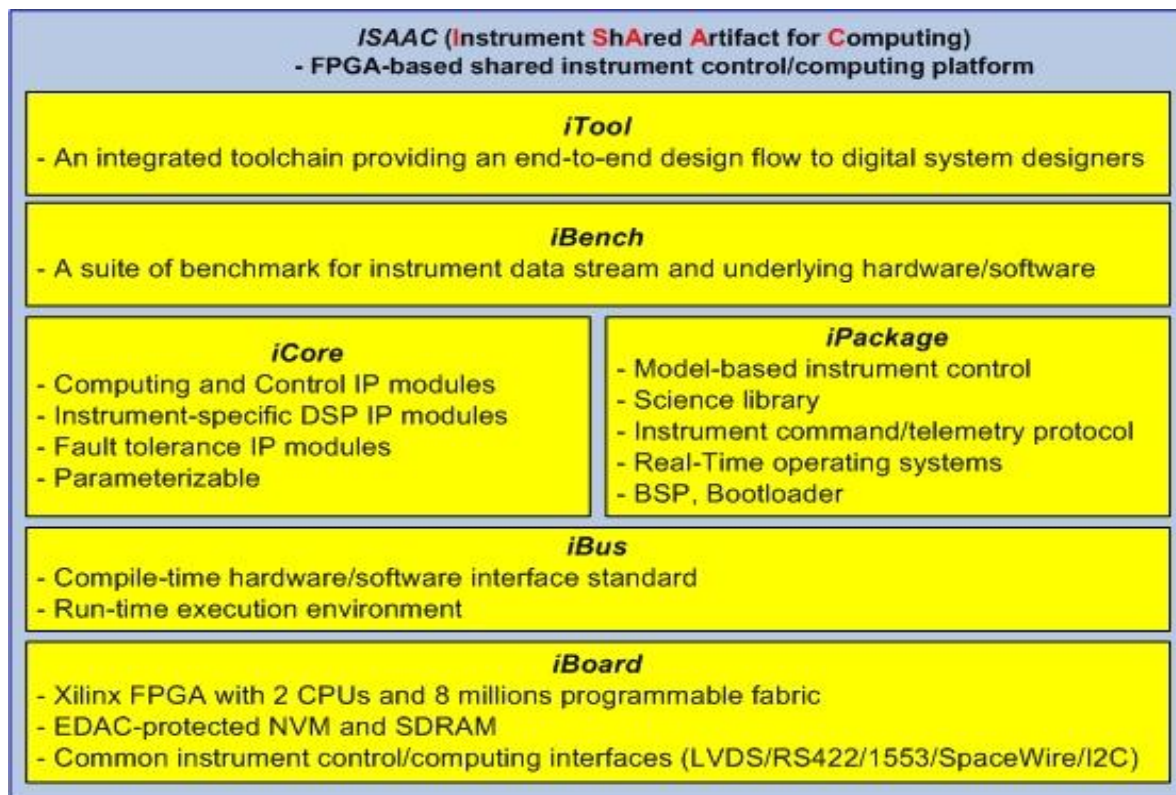


Figure 1: 6 sub-parts that make up ISAAC

	Isaac Newton (Radar)	Isaac Asimov (Imager)	Isaac Stern (Radiometer)
Mission Impact	SMAP Reduce downlink data rate	MSPI Data Reduction	HAMSR MBE-based Controller Instrument-S/C Interface
iBoard	ML410	ML410	ML410
iBus	CoreConnect	CoreConnect	CoreConnect
iCore	I/Q Demodulation & Decimation Filter	Bessel Function Fitting (Data Compression)	I2C Instrument Control & Timing
iPackage	BSP Linux, ICTP	BSP GSL, ICTP	BSP Linux, MBE, ICTP
iBench	Input: Memory Output: Matlab	Input: Hw generator Output: datafile	Input: Memory Output: iPanel/ICTP
iTool	ISE/EDK GNU SDE AccelDSP, TMRTTool	ISE/EDK GNU SDE ImpulseC	ISE/EDK GNU SDE

Table 1: Implementation of ISAAC

ISAAC is an R&TD project whose objective is to provide a standard FPGA based computing and control platform for various types of instruments. There are six sub-parts that make up ISAAC. They are iTool, iBench, iCore, iPackage, iBus, and iBoard. Starting from the bottom of Figure 1, there is the iBoard, which is the hardware substrate that includes the Xilinx FPGA, a few standard hardware interfaces for external instruments and a few analog-to-digital converters for gathering telemetry. The iBus is a standard and unified interface that links the hardware to the software to provide easy adaptability and integration of modules during the system configuration and operation. Above the iBus is the iCore, which holds all the libraries of RTL codes which are like machine/hardware code. Depending on the user's application, they may choose to configure which libraries to include. The iPackage is much like the iCore except that iPanel hold libraries for the software such as instrument control software. The iBench is a compilation of data that may be used for benchmark, verification, validation, and calibration of the instrument. Finally, the iTool is a tool chain that provides smooth design flow from front end algorithm development to RTL implementation on FPGA device.

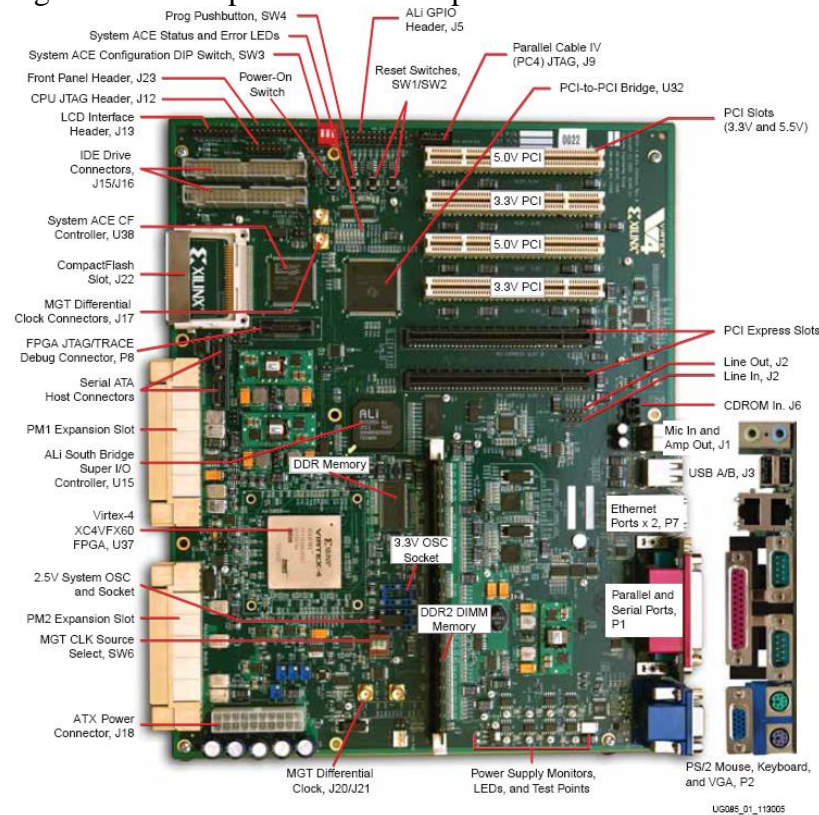


Figure 2: Prototype iBoard - Xilinx ML410
From: <http://www.xilinx.com>

ISAAC is currently developing control systems that will support instruments in the imaging, radar and radiometer field. The three projects demonstrating these capabilities are ISAAC Newton, which will demonstrate ISAAC's capabilities with radar instruments, ISAAC Asimov, which will demonstrate it's capabilities with imaging instruments and ISAAC Stern, which will demonstrate ISAAC's capabilities with radiometer instruments. Table 1 shows the different ISAAC projects and how they each utilize the six sub-parts that makes up ISAAC.

iCore Catalog

As mentioned earlier, iCore is storage for all the library of RTL codes. Figure 3 shows a breakdown of what RTL modules will be inside iCore. Currently iCore will support three types of instruments which include imaging, radar and radiometer instruments. The first task for iCore

was to construct a breakdown diagram that will help with getting a better understanding of how iCore is being implemented. The diagram will also speed the design process when building iCore because everything that needs to be in iCore has been thought out and sorted into its respective group.

iBoard Timing and Mapping for Digital Filtering Core

The task for the iCore sub-part was to verify how much resources the iBoard was using. This was done through a Linx machine that connected to the iBoard. The first thing that needed to be done was to check-out the program package that will be compiled onto the iBoard. The ISAAC team uses Subversion (SVN) to manage its repository and uses SVN to check-out files. A check-out in SVN allows the user to copy the desired file into a specified directory for the user to modify. After check-out, the next step was to compile the code and run simulations to check if the RTL design is correct. Once the RTL design has been verified, the next step is to do the synthesis and Place-n-Route. In this step, the RTL code is compiled and translated into a format that can be used to implement a Xilinx FPGA. After the RTL code has been compiled, there is a 10 – 30 minute wait time for the Place-n-Route to finish. The Place-n-Route process is a process that takes the translated RTL code, reads the code, and configures the FPGA accordingly. After the Place-n-Route is finished, the next step is to verify the FPGA by checking its timing and mapping. There are two types of FPGA that the iBoard uses and they are Xilinx Virtex 2 and Xilinx Virtex4. For the verification, the numbers of multipliers, slices, flip-flops and look up tables (LUT), along with how fast the processor runs needs to be determined. Table 2 shows the end results of determining the timing and mapping of iBoard.

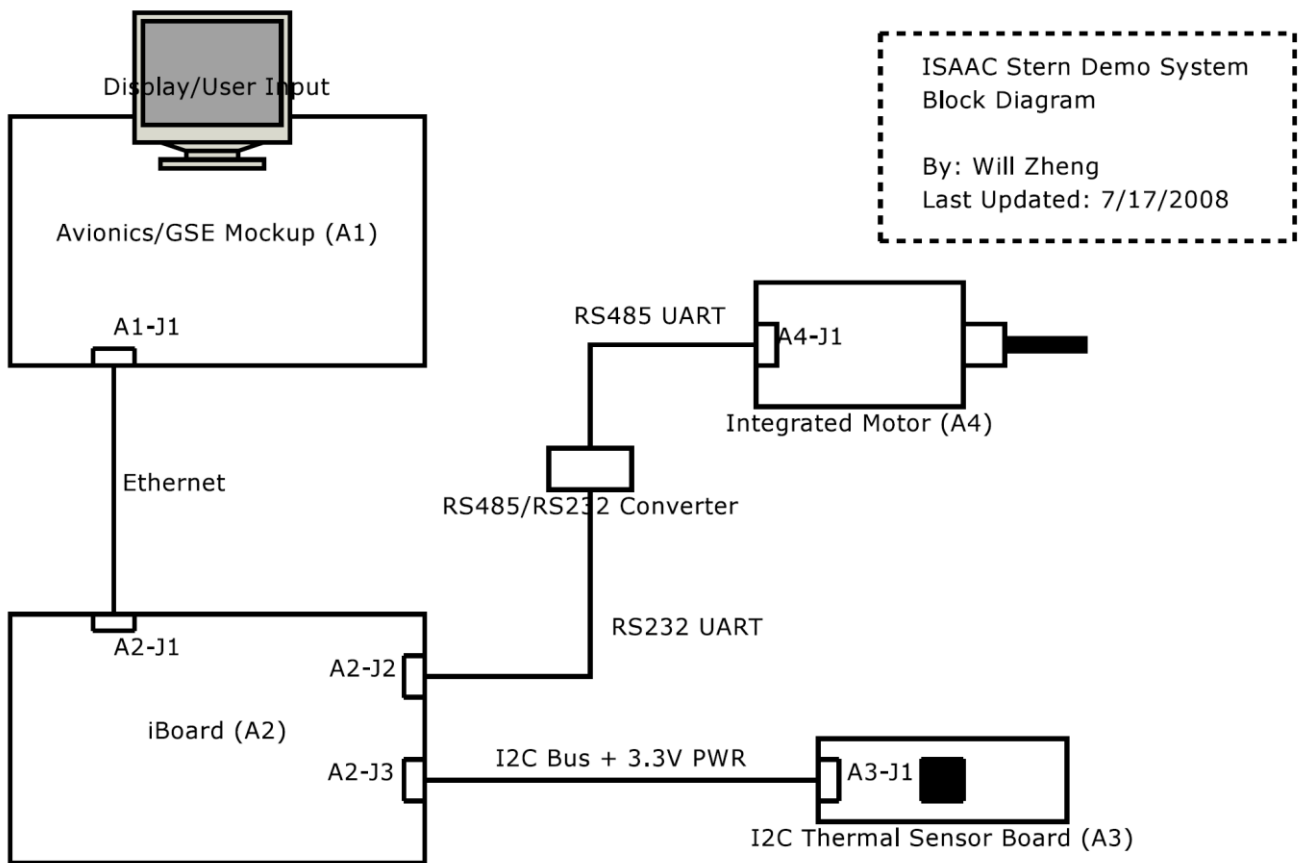
PARAMETER:	Xilinx Virtex 2: speed grade 4		Xilinx Virtex 4: speed grade 10	
Speed (Bandwidth)	91.4MHz		112.3MHz	
	Used / Total	Percentage %	Used / Total	Percentage %
Number of Multipliers	36 / 96	37%	36 / 128	28%
Number of Slices	10,498 / 14,336	73%	10,533 / 25,280	41%
Number of Flip Flops	15,753 / 28,672	54%	15,776 / 50,560	31%
Number of 4 input LUTs	11,910 / 28,672	41%	11,817 / 50,560	23%

Table 2: iBoard Verification

Motor Control

In order to keep the ISAAC project going, a demonstration was planned showing that it is possible to gather telemetry through iBoard. It was decided that telemetry would be gathered from a UART motor and an I²C thermal sensor. Figure 4 shows a block diagram of the demonstration system.

The motor chosen for the demonstration is a Lin Engineering SilverPak 17CE motor. This motor is a bipolar stepper motor with integrated controller, driver, and encoder. To learn how the motor work, it was first connected to a UART RS485 to RS232 converter that was connected to the laptop. The user could choose to use the GUI software provided by the company or use HyperTerminal to send commands to the motor. Some of the problems that were encountered happened during setup. At first, the green LED that indicated that the motor was on didn't appear but it was later discovered that there was not enough current being supplied to the motor. Another problem that was encountered was that once the green LED indicator turned on, the motor would not spin. The problem was that the step size being inputted to the motor was too small; the motor needed an input of a step size larger than what was stated in the manual. The



next step will be to implement the motor to the iBoard.

Figure 4: Demo System Block Diagram

Thermal Sensor

The thermal sensors for the demonstration are able to interface with the I²C bus on the iBoard. The thermal sensor chosen was the MIC184 I²C/SMBus thermal sensor. Before testing the thermal sensor with iBoard, it was tested on a PC laptop through an iPort/AI that connected to a RS232 connector. The iPort/AI is an I²C host adapter that connects from the computer to an I²C device. One of the major problems we had with the MIC184 was getting acknowledgement from the thermal sensor. According to the datasheet, the slave address is supposed to be 48h (in hexadecimal) however when calling that address in HyperTerminal, there is no response from the thermal sensor. It was discovered that for some reason, while using HyperTerminal, the user must left shift the address bit. Thus if the user wanted to talk to the slave whose address bit is 48h, the user has to convert 48h into binary, left shift and convert back to hexadecimal. The correct address bit to for the MIC184 while using HyperTerminal is 90h. After the address bit had been changed, the thermal sensor was able to acknowledge the master I²C bus. However, the readings on the thermal sensors were not accurate. According to the company, the thermal sensors did not need any calibration and the sensor should be reading the ambient temperature.

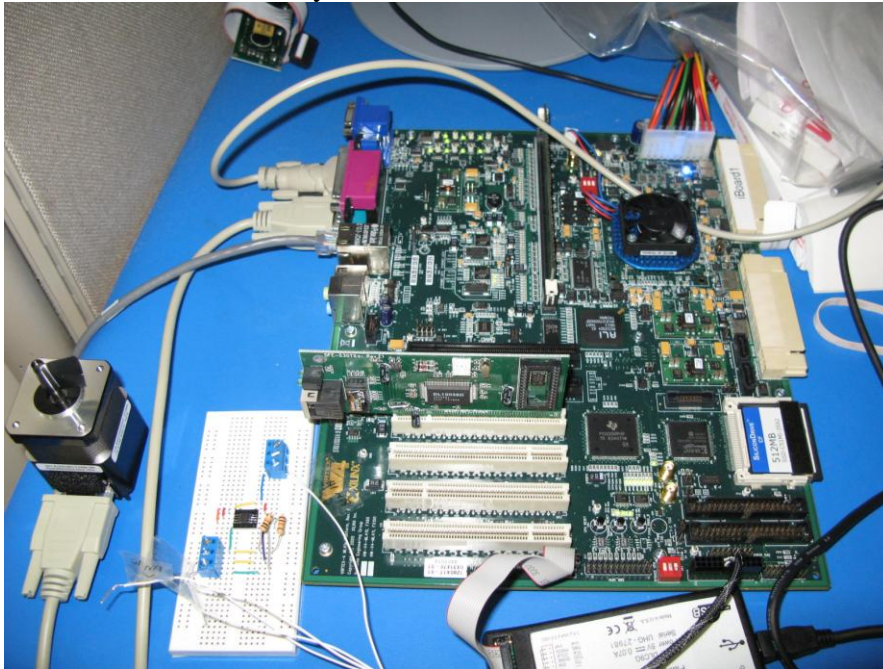


Figure 5: Actual Demo Setup

The MIC184 thermal sensor was found to be reading four degrees Celsius higher than ambient temperature after verification with a thermometer. Trying to read the temperature as closely as possible, another temperature sensor was ordered; LM75. Again, same as the MIC184, the address bit had to be shifted left in order to use HyperTerminal. The readings on the LM75 were also inaccurate however when the ambient temperature

reached above a certain point, the LM75 was found to be accurate within $\pm 0.5^{\circ}C$. The LM75 was chosen to be the thermal sensor that will demonstrate the iBoard's ability to gather telemetry data.

Implementing motor and thermal sensor

Figure 5 shows the actual demo setup with the stepper motor and thermal sensors connected to the iBoard. Both the motor and the thermal sensor have been tested separately with a PC laptop and proven that they both work with the exception that the thermal sensor's reading was not accurate. The thermal sensor has been tested on the iBoard and proven to work however

the motor has not been tested yet with the iBoard. The problem now is that the connection with the motor and iBoard has not been established. Once a connection has been established, it will be possible to demonstrate the ability of iPackage. The demonstration consists of gathering telemetry from the thermal sensor. If the thermal sensor reads a temperature above a set threshold level, the iBoard will turn on the motor and start gathering telemetry from both the motor and the thermal sensor. The telemetry will then be sent to the host computer that will be displayed on the iPanel which will be discussed in the next section.

iPanel

Another part of iPackage is to provide the user with a graphical user interface (GUI) that will allow them to control the on board instruments remotely. In addition to being able to send commands to the instruments remotely, the user will also see telemetry data coming off the instruments. Figure 6 shows a flow diagram of how iPanel is intended to work for the demonstration. The program used to create the GUI was Python/Tkinter. In this demonstration, the telemetry coming from the LM75 thermal sensor along with some benchmark data and MBE (model based engineering) from the iBoard will be sent to the host computer. From the host computer, the data coming from the iBoard will be made into well-defined, standardized, data structure that will be passed along to the front-end display.

Figure 7 shows the envisioned design for iPanel's layout and Figure 8 shows the actual design of iPanel. iPanel allows the user to select an instrument to be controlled. When an instrument is selected, the user will be able to see the state of the instrument and resource usage. The user can select which telemetry they would like to track in a graph form and will also have the current telemetry displayed in a list form. The user can also control the instrument by sending commands in the command console in iPanel.

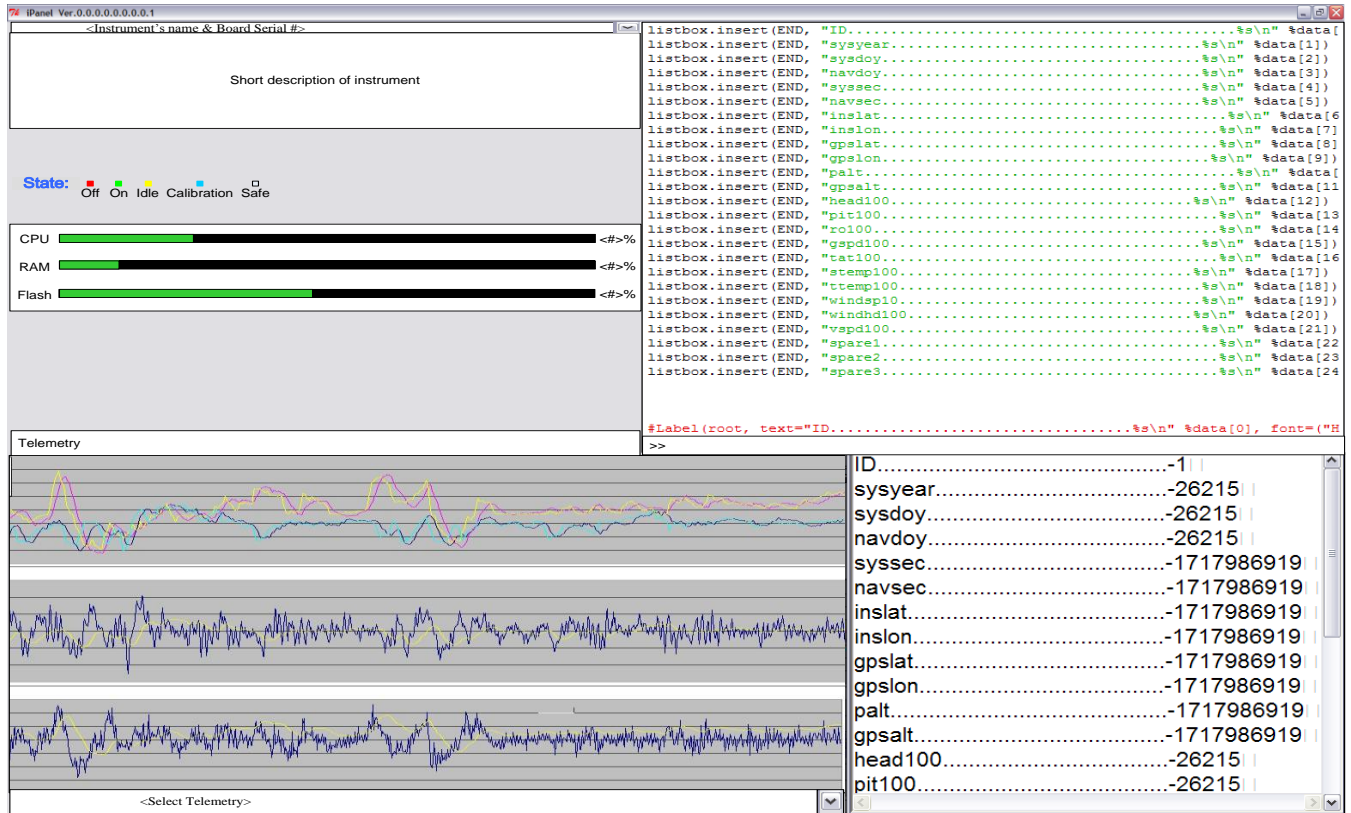


Figure 7: Envisioned design of iPanel

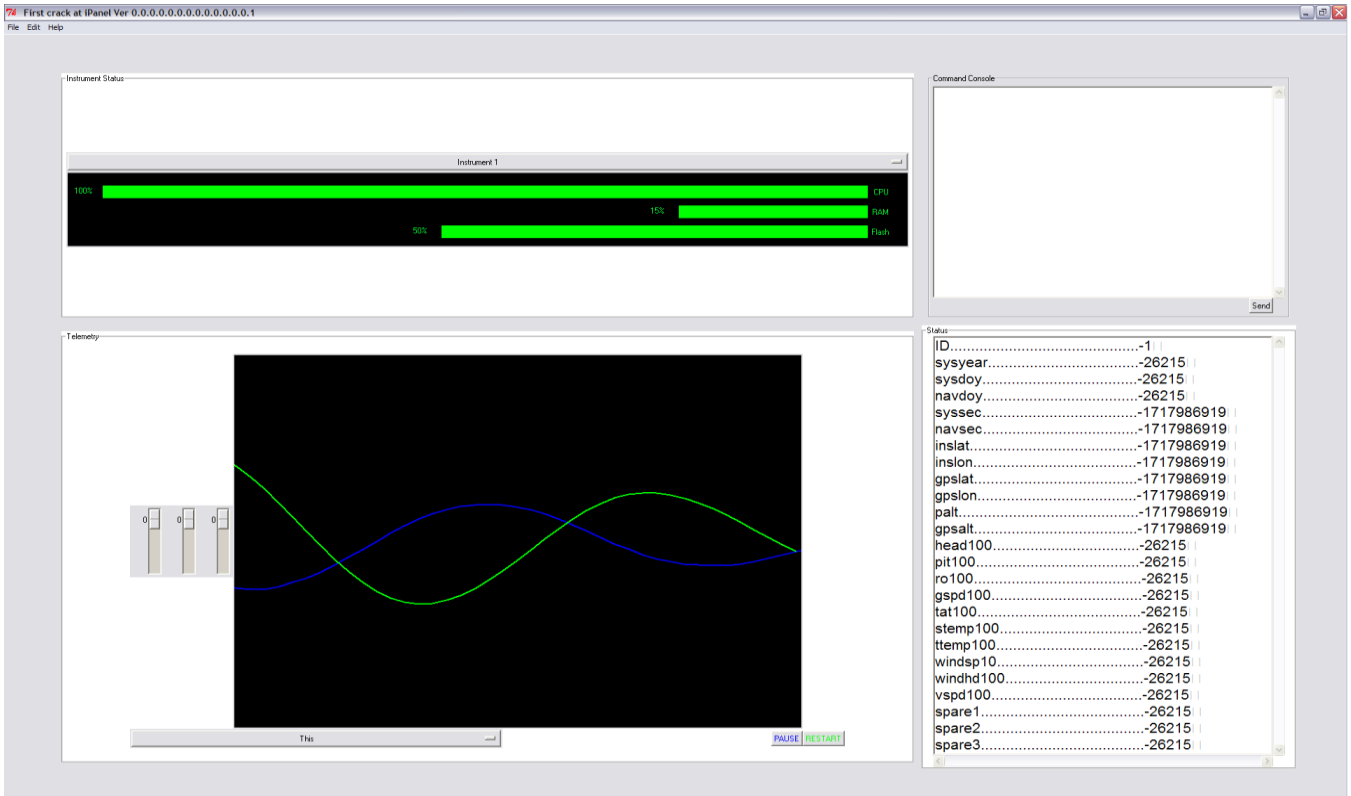


Figure 8: Actual design of iPanel

Contributions

1. Redesigned iCore catalog in Viso
2. Verified resource usage for the digital filtering core on iBoard
3. Ordered thermal sensors and stepper motors
4. Soldered and designed thermal circuit
5. Tested thermal sensors and stepper motors
6. Setup iBoard demo system
7. Designed iPanel GUI

CONCLUSION

This project required a large learning curve time period. Majoring in electrical engineering and concentrating in signal processing, embedded systems was a new thing that had to be learned. A lot of time went into learning the technical terms and acronyms in this field but not only understanding what they meant but how they work and why it was important. A new thing that had to be learned for this project was the programming language Python and TKinter. Learning how to use Linux didn't require much learning curve however; new techniques to enhance the Linux experience were learned. The next step for this project is to make a fully functional iPanel GUI display that is able to gather data in real time and send commands to the instruments. Also, although the thermal sensor and motors have been proven to work on the PC laptop and iBoard separately, they still need to be tested on iBoard together.

ACKNOWLEDGEMENTS

Dr. Yutao He was a great mentor. We had tag up meetings and quiet hours every week to make sure that if I had questions, I could ask. Also, even when he was busy, if I had a question, he would take some time to help me. Elizabeth Alfon and Debra Cuda were always there to help with JPL or office needs. They were very helpful in making things happen as quick as possible and always made sure everything that was needed was there. Will Zheng, Jason Zheng, and Kayla Nguyen helped with the project. They were like my mentors when I couldn't get a hold of Dr. He. Geoffrey Vaughan and William Jay were great officemates. If I had a question about JPL, they were more than happy to answer; they also helped me search for jobs at JPL and took me out to lunch every week. I would also like to thank Lindsey Holland, my co-intern, for helping me out in general.

REFERENCES

"FPGA vs. ASIC" [Online]. Available:

<http://www.xilinx.com/company/gettingstarted/fpgavsasic.htm>

"FPGA Basics" July 25, 2008. [Online]. Available: <http://www.andraka.com/whatisan.htm>

Z. Junaid, "FPGA: The Chip That Flip-Flops" Oct. 1, 2004. [Online]. Available:

http://209.85.173.104/search?q=cache:Fs2Kkub_FDwJ:www.cs.fredonia.edu/zubairi/training/fpga.ppt+why+FPGA&hl=en&ct=clnk&cd=1&gl=us