Set-Top Box Simulator

Senior Project Report

Author & Developer: Philip Tyler
philip.b.tyler@gmail.com

Sponsor: TiVo Corp.
www.tivo.com

Advisor: Christopher Lupo
clupo@calpoly.edu

California Polytechnic State University
San Luis Obispo
June 7, 2013

Table Of Contents

I. INTRODUCTION

1. Abstract

This report presents a python-based Set-top box simulation program utilizing a Simulation library called SimPy (See Appendix 1) to simulate real-time operation of a Set-top Box, or DVR. A graphical user interface, designed with PyQt4, allows a user to customize many simulation parameters such as hard drive speeds, buffer sizes, length of simulation, etc. The GUI also shows the user any errors that occur during the simulation such as buffer overflows/underflows. The results of this simulator lie within 85%-95% accuracy depending on the user-input parameters. With this simulation program, a Set-top box hardware or firmware developer can interchange the scheduling algorithms and simulation parameters to find the ideal system to manufacture.

2. Terminology

ATR: (Adjacent Track Repair) A self-maintenance function of modern HDDs. When a small group of tracks is rewritten a few times, the HDD will flush its cache memory and check the grouping for errors, rewriting where necessary. This causes a small break in HDD processing time.
Benchmark: A time measurement of a task, run from start to finish, within a known system.
Buffer: A region of a physical memory storage used to temporarily store data while it is being moved from one place to another.
Buffer Watermark: A defined amount of used memory in a buffer. When the amount of valid bytes in the buffer exceeds (high watermark) or precedes (low watermark) the defined amount, the system is warned the buffer might overflow or underflow.
DVR: (Digital Video Recorder) A Set-top Box specifically designed for consumer recording of cable/satellite channels.
GUI: (Graphical User Interface) Allows users of a program to interact with images rather than text commands.
Header: The small parts of a disk drive, that move above the disk platter and transform the platter's magnetic field into electrical current (read the disk) or vice versa – transform electrical current into magnetic field (write the disk)
HDD: (Hard Disk Drive) A data storage device used for storing and retrieving digital information using rapidly rotating discs (platters) coated with magnetic material.
IDE: (Integrated Development Environment) A software application providing software development utilities for computer programmers, usually consisting of a source code editor, syntax highlight/code-completion tools and a debugger.
Python: A dynamic programming language and interpreter used in a variety of application domains with features like: clear readable syntax, large built-in library and full modularity.
PyQt4: Python v2 and v3 library for Digia's Qt GUI application framework and

runs on all platforms supported by Qt including Windows, MacOS/X and Linux.

Set-top Box: An information appliance device with a tuner connected to a television, turning the source signal into content in a form that can then be displayed on the television screen or other display device. They are used in cable television and satellite television systems, as well as other uses.
SimPy: An object-oriented, process-based discrete-event simulation language for Python, released under the GNU Lesser GPL (LGPL) license.
SIP: A tool to create Python bindings for C and C++ libraries, originally developed to create PyQt.
Tuner: A component of set-top box hardware. Converts a radio frequency analog or digital television transmission into audio and video signals which can be further processed to produce sound and video output.

3. Problem Statement

After joining TiVo Corp. as an intern summer of 2011, the engineers were designing the next generation TiVo DVRs. DVRs with more channel tuners/ recorders, higher resolutions and a multitude video outputs. However, the current single Hard disk drive minimal architecture bottlenecked the set-top box. The single HDD could handle the constant stream of media data reads and writes, but only if they were processed in the most efficient scheduled order. Some other architecture features brought into question were buffer sizes, size (in MegaBytes) of HDD read/write requests and what HDD to choose.

Instead of designing the multiple hardware schematics, spending tens of thousands of dollars ordering the prototype boards, then running countless time-consuming high-stress tests with different HDDs and scheduling algorithms our team, the hardware team, set out to build an all-inclusive Set-top box simulator.

II. BACKGROUND

1. Initial Project Guidelines

The main goal of this project was to create a design tool for the TiVo engineers. A flexible and quick software tool for testing purposes. At the time, TiVo was preparing for the release of their newest DVR, the TiVo Premiere XL the first TiVo set-top box capable of recording four channels at once. Months of beta testing the new equipment yielded results of high HDD traffic. The engineers had plans of adding more tuners and media output streams to the next generation of DVRs, but not enough HDD bandwidth with the current systems in place.

The tool would allow TiVo engineers to test their latest hardware and HDD scheduling improvements against a user-modified virtual set-top box simulation. The simulator engine would be fine tuned to exact timing values found in results of real-time hardware testing. The tester could enter system parameters such as buffer sizes, scheduling priorities and input/output bitrates then let the simulation run for however long. While running, the simulation could collect errors, warning and log messages the tester could read through after completion.

2. Customer Requirements

The Hardware team at TiVo needed a simulator with the following features:

A. User-editable parameters including
   i. HDD: Cache size, Max/Min R/W speeds
   ii. HDD: Time to move HDD head one track, from first to last track and rotate the disk 360 degrees (all physical test-verified times)
   iii. Chance of Adjacent Track repair or similar HDD self-maintainence
   iv. Time to run simulation.
   v. Size of media buffers, separate values for input/output
   vi. Number of input channels with corresponding average bitrate
   vii. Number of output channels with corresponding average bitrate
   viii. High and Low watermarks for buffers
   ix. Max MB-size of read/write HDD requests
   x. HDD request timeout to generate request deadlines
   xi. CPU processing time
   xii. RAM read/write time

B. A Graphical User Interface, or GUI, to show
   i. Dialog for user to edit parameters above
   ii. Error messages
   iii. Log messages
   iv. Used space in buffers
   v. Progress Bar running simulation
   vi. Interface to start/stop/reset simulator

C. The application had to be portable and easy to use
   i. Packaged for easy transfer
   ii. Ready to install on any operating system
   iii. Simple interface so any employee could use it
   iv. Easy to understand code for easy manipulation

3. Engineering Requirements

The above customer requirements were converted to engineering requirements:

A. A programming language with the following features:
   i. Simple installation on (almost) every OS
   ii. Large selection of advanced GUI libraries
   iii. Expansive Modular simulation package(s)
   iv. Availability of tutorials, well-written APIs
   v. Object Oriented
   vi. Open Source License

B. A Simulation Library With the following features:
   i. Simple Installation
   ii. Well-written documentation and tutorials
   iii. Open Source License
   iv. Minimal-to-no dependency on other libraries
   v. Time-based
   vi. Fully Modular
   vii. Object Oriented

C. A GUI library with the following features:
   i. Simple Installation
   ii. Well-written documentation and tutorials
   iii. Open Source License
   iv. Minimal-to-no dependency on other libraries

D. The Source code for the application must be
   i. Easily readable so future developers and team members can quickly understand code
   ii. Commented where necessary to explain more complex algorithms
   iii. Utilize Object-Oriented properties for application expansion and Polymorphism
   iv. Able to interchange scheduler and HDD classes for different Set-top box simulations

4. Design Decisions

Python over Java and C++

After discussion with other developers, Python became the language and base-layer of the software. The choices were Python, Java or C++. C++ had quick execution time, but a steep learning curve with odd complexities. Java met all the engineering requirements, but other developers were concerned about the performance of the Java Virtual Machine (JVM) and few simulation libraries. Python stood out as the perfect fit.

Python's readability, extensive standard library with thorough API documentation and tutorials provided an easy learning curve. It's licensed as open source and rich with thousands of open source packages for many application domains, inlcuding simulation and GUI.

SimPy for simulation

The director of the Hardware Team, David Platt, suggested SimPy as the simulation skeleton library. After some research, it met all requirements with great reviews from other developers. SimPy utilizes a process-based, discrete-event simulation library with classes for Processes, Resources, Stores, Monitors, Levels, Interrupts and Events. The developers provide a detailed API and tutorials to get developing quickly. SimPy even included its own GUI.

GUI Built with PyQt4

During the first few months of development, the GUI was programmed with SimPy's GUI. However, its graphical architecture could not handle constant refreshing to update the progress bars representing the used data with in a simulation data buffer. Two other Python GUI libraries were praised by other developers online and in-office: wxPython and PyQt. The documentation and capabilities of PyQt far exceeded that of wxPython, making it the obvious choice.

The final GUI was built with PyQt4.2, the most recent build of the Python port of Digia's Qt GUI framework developed in C++. It's open source, and used in hundreds of applications. The most impressive library feature, Signals and Sockets, provided a simple architecture for connecting GUI elements such as buttons, dials and textboxes to python functions.

Development Tools

The first decision involved choosing a strong Python IDE to edit, debug and consolidate the application. The development program had to be compatible across all OS environments for the convenience of future developers. Must be feature-rich with extensive text-editor preferences.

The team decided on the Eclipse IDE with the PyDev plugin. The team had previous experience using Eclipse for C++ and Java development on Linux, OSX and Windows operating systems. Both the IDE and plugin were open source and feature-rich. Initially the project was developed on

III. DESCRIPTION

1. System Requirements

In order to run the simulation on a desktop or laptop computer, the following preconditions must be met:

A. The computer itself must meet Python v2's system requirements
   i. Windows
      a. Windows 2000 or newer
      b. Cygqin interpreter (optional)
   ii. Macintosh
      a. OS X (any version)
   iii. Linux/Unix
      a. Kernel version 2.5 or later
      b. Distribution must have GUI such as Gnome
   iv. Hardware minimums
      a. 256 MB RAM
      b. 500MHz processor

B. These software packages must be installed
   i. Python 2.6 or 2.7 interpreter (not tested with Python 3)
   ii. SimPy 2.3 (latest release)
   iii. SIP 4.14 (latest release)
   iv. PyQt 4.10 (latest release)

2. Simulator Engine

Class Descriptions

The SimPy library is Object Oriented and includes generic classes for Processes (repeated tasks like scheduling HDD requests), Resources (modules needed for a process to run) and Levels (a containing that can be filled or drained like a data buffer). SimPy is a time-based simulation package for python. It utilizes a queue of processes ordered by when they can run next. Processes call 'yield' to delay processing for a set amount of time or until they receive a requested resource. The specific simulation classes below extend the generic classes a specific modeling purpose.

A. Input/Output data streaming model
   i. Buffer - subclass of Level to model a single unit of capacity-defined memory.
   ii. Buffer Control - subclass of Process to model kernel checking level of Buffer and determining if a HDD request should be made. Can only have one open HDD request at a time.

a. IO Request - Class representing a HDD request with properties such as read or write flags, byte size, a deadline, and origin buffer.
b. Read Requester - subclass of Buffer Control set to monitor an output buffer. If the buffer contains less data than its low water mark, or it has not created a HDD request for some time.
c. Write Requester - subclass of Buffer Control set to monitor an output buffer. If the buffer contains less data than its low water mark, or it has not created a HDD request for some time.

iii. Data streaming
a. Input Stream - subclass of Process to model a cable tuner constantly filling a data buffer with demodulated audio/video data.
b. Output Stream - subclass of Process to model an MPEG decoder chip constantly emptying a data buffer's MPEG data and converting it to sound and video signals ported to a display (television)

B. HDD scheduling model
i. Scheduler - subclass of Process with abstract function definitions
a. TiVo engineers create subclasses of Scheduler with specific algorithms.
b. These specific subclasses must implement the abstract methods called by other modules in the simulation. If the functions to no match the parent class, the specific scheduler will not be run properly.

C. HDD model
i. Disk - subclass of Resource containing user-entered HDD properties.
ii. HDD Cache - subclass of Process to pull HDD requests off the schedulers sorted queue when the cache is not full. Request in the cache enter another queue, waiting to be written to/read from the HDD.
iii. HDD Controller - subclass of Process to perform HDD read/write request simulation.
a. After performing a write request the HDD Controller may enter ATR mode, stopping processing of HDD requests. The chance of ATR state is given by the user.

D. Other System Models
i. CPU - subclass of Resource to model a single-core processor
a. Acts as a mutex that Process subclasses must have possession of in order to simulate calculations and processing.
b. Mostly used by scheduler when sorting HDD requests
ii. RAM - subclass of Resource to model the time required to mov data between buffers/caches.

Simulator Flowchart

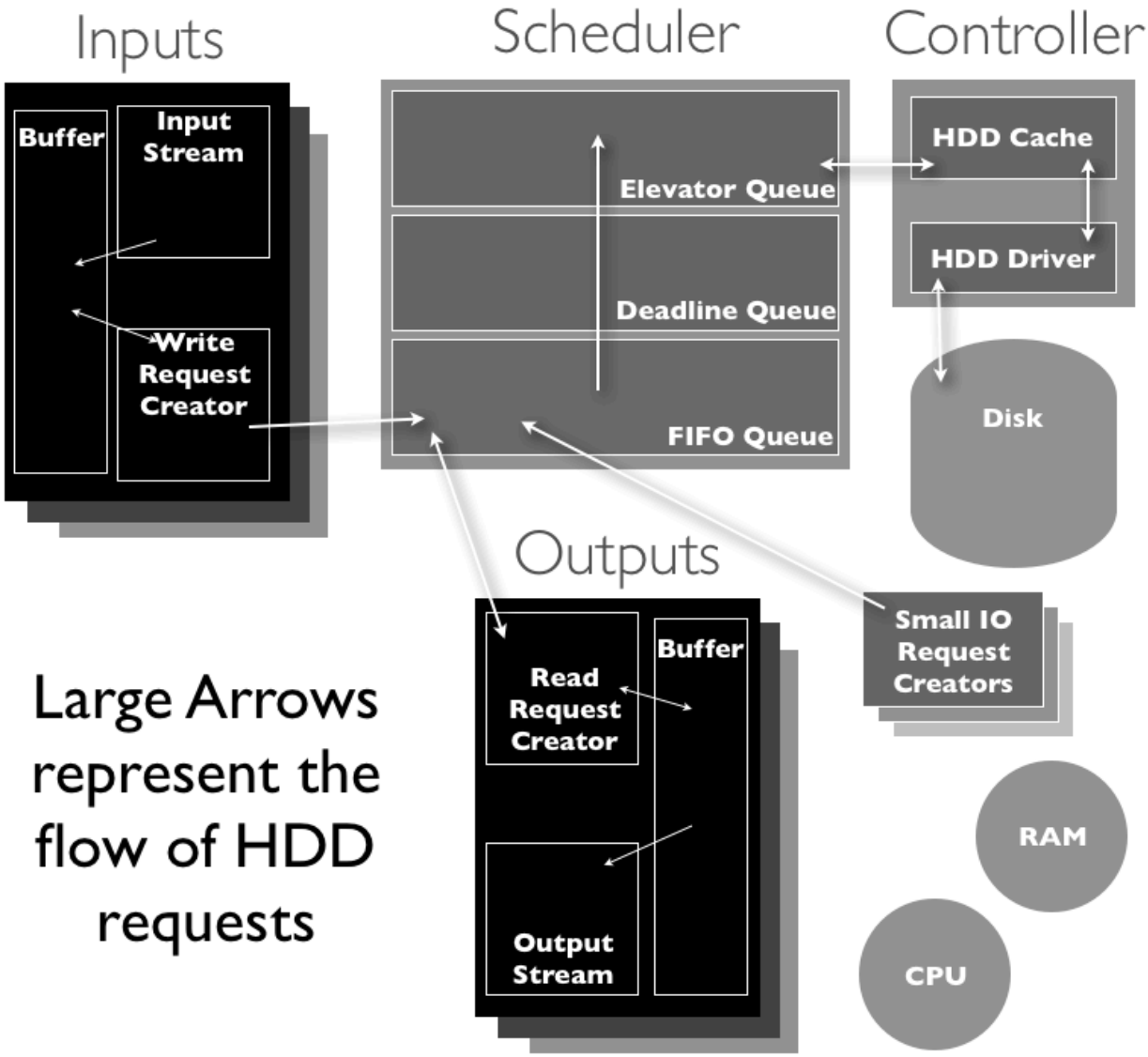

Figure 1: Diagram of all simulator components

User Editable Parameters

Table 1 illustrates all the simulation parameters the user can change to affect the simulation environment. Note that millisecond values are simulation milliseconds, not real-time.

Variable Name	Default value(s)	Description
numTuners	6 input streams	Number of inputs streams and buffers the simulation uses
iBitrates	20 Mbps	Average data per second written into all input buffers
iBufferSize	8000 kBs	Size of input buffers
numOutputs	4 output streams	Number of outputs streams and buffers the simulation uses.
oBitrates	20,20,20,20 Mbps	Average data per second read from all output buffers
oBufferSize	8000 kB	Size of output buffers
iInterval	50 ms	Time between input buffer checking its used space to make a write request.
oInternal	50 ms	Time between output buffer checking its used space to make a write request, in simulation milliseconds.
oThreshhold	oBufferSize/2	Maximum amount of data needed in an output buffer to not create read requests.
iReqTimeOut	1000 ms	Max time input buffer will wait before making an IO request of the largest size possible
oReqTimeOut	1000 ms	Max time output buffer will wait before making an IO request of the largest size possible
wMaxSize	2048 kB	Max size of HDD write request
rMaxSize	2048 kB	Max size of HDD read request
cpuTime	2 ms	Time yielded for a single unit of CPU processing such as scheduling lists, moving memory, ect.
maxTime	5000000 ms	Time simulation will run for in simulation milliseconds.
sIOPeriod	100 ms	Average time between small IO HDD queries, in simulation milliseconds. To model kernel system calls.
sIOMaxSize	5 sectors	Max number of sectors small IO HDD queries will write.
sIOBurstTime	50 ms	Time between IO requests in one burst.
sIOMaxQueries	2 requests (in one burst)	Max number of MySQL IO queries sent after mfsPeriod.

Variable Name	Default value(s)	Description
sqlOPeriod	2000 ms	Average time between MySQL IO HDD queries, in simulation milliseconds.
sqlOBurstTime	20 ms	Time between IO requests in one burst.
sqlOMaxSize	4 sectors	Max number of sectors MySQL IO queries will write.
sqlOMaxQueries	5 requests (in one burst)	Max number of small IO queries sent after mfsPeriod.
hddNumTracks	100 tracks	Time to simulate the HDD processing a new command sent from the driver in simulation milliseconds.
hddMaxTurn	11.1 ms	Time for disk to make 1 full rotation. Used to generate time needed for adjusting the needle to a certain sector on a track.
hddTrackMove	1.63 ms	Time to simulate the HDD moving its head across 1% of all total sectors.
hddFullStroke	28.55 ms	Max time to simulate rotating the disk to the correct orientation in simulation milliseconds.
hddOneMedia	1.2 ms	Time to simulate the HDD writing one 128kB section of data to the disk.
hddCacheSize	16 MB	Size of HDD cache
hddHoldTime	100ms	Time to simulate repairing one hdd track.

Table 1: All simulation parameters, default values and descriptions

3. Graphical User Interface

The simulator's GUI is built with the PyQt4. The custom simulator GUI classes subclass PyQt's GUI objects such as QDialog, QMainWindow, QLabel, QStatusBar, etc. Each subclass corresponds to a specific element in the GUI, and contain functions and an overridden constructor. All the GUI coding and class declarations. Because Python allows multiple inheritance, some GUI components also subclass SimPy's Process class, thereby allowing them to be updated during simulation such as the error log window and buffer bars.

The following images display the two GUI windows (dialogs) for the simulation. *Figure 2* shows the main window containing the error log window, progress bars for each buffer representing the used space inside the buffer refreshed every tenth of a second, menu bar, status bar and an overall simulation progress bar. *Figure 3* shows the parameter editor dialog. Every parameter has a spinbox preloaded with default values. When the user clicks apply, the main window is redrawn with new parameters.

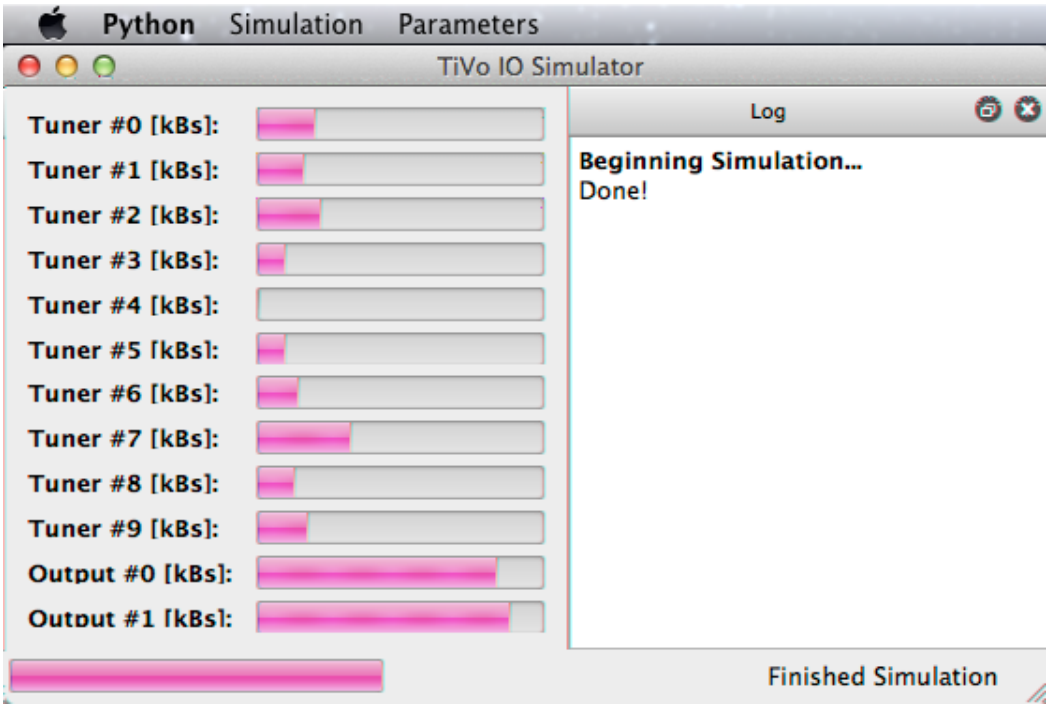

Figure 2: Main window for simulation GUI

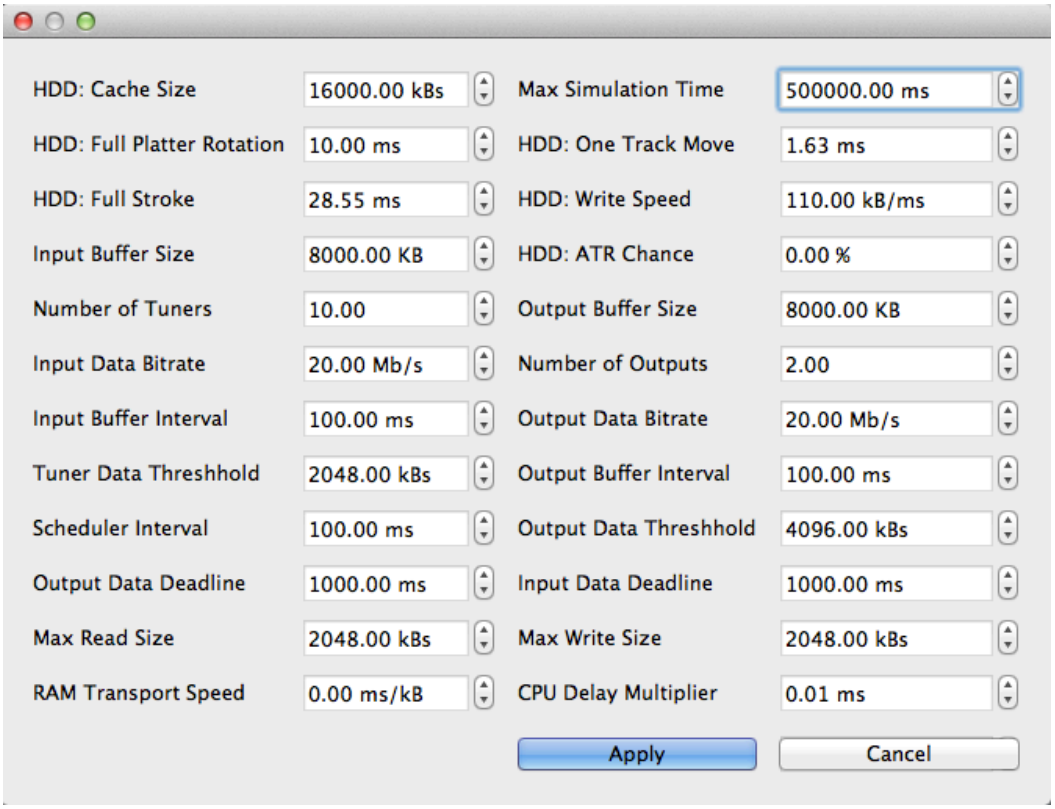

Figure 3: Simulation parameter editing window.

IV. EVALUATION

1. Testing

Testing for this software tool was done in two main ways. Code testing and debugging were done through the Eclipse Debugger and Terminal window. Larger scale feature and component testing occurred in simulator code reviews by the Hardware Team.

Eclipse Debugger and Terminal

Most of the testing process involved trail and error. In order to add a new feature to the simulator, the feature was split into phases. Each phase was implemented in the code, then tested before adding the next phase. For example, when coding a scheduler class, the first phase consisted of a function called by Request Creator objects to add new requests to be scheduled. A test was written with four Request Creator objects set to send a request to the new scheduler object, then delay for a pseudo-random amount of time. The correct scheduler operation was verified using The Eclipse Debugger to visually display object properties and the Terminal window to display log/error messages. The entire course of the project utilized this basic testing strategy

Code Review Meetings

Every two to three weeks, the team would meet up for a simulator review. The meeting would begin with a demonstration of the newest simulator features and changes since the previous meeting. The team would discuss the accuracy and capabilities of the new features, possibly suggesting different algorithms or tweaks for improvement. A conversation on future development of the simulator followed the code review session, resulting in a prioritized list of new features. Between meetings the new features would be added in priority order.

2. Verification

One of the most important aspects of the simulator was to yield accurate results, closely matching actual set-top box benchmarks (see *Terminology*). Once the core modules of the simulation components operated to spec, the next step was to run simulations with parameters matching known set-top box systems. Then verify the simulator results with predetermined hardware-tested benchmarks, bandwidth tests, and manufacturer-supplied timing diagrams. If the simulator could not reproduce matching results for known timings, its operation as a pre-design tool would be highly inaccurate.

Supplementing Benchmarks for Estimations

In implementing the RAM component of the simulator, the timings used for memory transfer delay calculation came from the timing diagram of the RAM inside the newest TiVo DVR model. Previous RAM benchmarking tests produced much longer memory transfer and overhead time. The team agreed those numbers' inaccuracy could be a fatal flaw in the performance of the SIM. More C++ benchmarking tests, specifically for memory, were written and run on other DVRs with known hardware. The constants, used for SimPy Process delay were adjusted until the simulator's known system results matched the tested benchmarks.

Better HDD Modeling

The HDD proved the hardest component to model, with many inner components including cache memory, firmware and magnetic heads to write onto disks. First HDD simulator results showed much higher maximum data throughput compared to previous set-top benchmarks by almost 20%. Even after reducing the HDD read/write speed simulation parameter from 110MBps (from Wester Digital Datasheet) to 88MBps (20% less), results still showed more data throughput when compared to benchmarks. Further modeling of the HDD's internal maintenance was needed.

Adjacent Track Repair (ATR), Western Digital HDD's disk-error checking and correcting process, runs after four adjacent tracks called a Zipcode is written over on separate occasions. Because of the close proximity between tracks, potential for an accidental bit change is high. After speaking with the Vice President of HDD Testing with WD, much more details of ATR were revealed. ATR can block any HDD read or write traffic for up to 130 ms while checking and possibly repairing the four adjacent tracks but usually averages about 100 ms. Once this feature was implemented in the simulation, simulator results matched benchmarks between 85-95%.

3. Trouble Shooting

Infinite Loops

Infinite loops caused the majority of errors observed. The GUI and terminal would stop responding and require kernel-issued termination. In such cases, breakpoints were placed in the code, and the Eclipse Debugger would step single lines of code until the relentless loop was identified and removed. GUI infinite loops presented challenges because the infinite loops were found within the Library's code, requiring a backtrace to find the incorrectly assigned variable(s).

The Rebuild

The first iteration of the project completed after 3 months of development at TiVo Corp. It used SimPy's built-in GUI framework, very few Object Oriented design principles and full of debug print statements. The team agreed to rebuild the project from scratch, utilizing PyQt4 GUI, Inheritance/Polymorphism on buffer control and scheduler Python modules and cleaner code. The second iteration was completed August 2012, a year later.

4. Fine Tuning

The final additions to the simulator were GUI finishing touches. First, Adjusting minimum and maximum parameter values. Ensuring when the minimum or maximum parameter values were used, the simulator still produced realistic results. Other minor fixes included correct progress bar response, adding reset/redraw functions to the menu bar, and formatting log files.

5. Results

The final results of simulator testing proved simulations with parameters matching those of two known the systems, the TiVo Premiere and TiVo Premier, came within 95% accuracy for maximum HDD data throughput and timing benchmarks. Further result details cannot be contained in a non-TiVo Corp. document. With known system accuracy like this, the tool proved accurate enough for future set-top box development simulation.

V. CONCLUSION

1. Project Summary

In conclusion, the results of the software tool's simulation results matched actual hardware test results to provide a Set-top box design tool for TiVo engineers. The final design is completely cross platform thanks to Python and its libraries' compatibility. With a simple GUI and an array of editable parameters, almost any member of the company could install and operate the simulation. The well documented Python source code enables any future simulation developer to continue additional editing of the software.

After the testing, verification and fine tuning processes the simulator yields results 85% - 95% accurate results, when compared to equal real-time hardware benchmarks. The closer the simulation parameters are the actual hardware benchmarked, the more accurate results. In simulations where many parameters vary far from those in the physical benchmark tests, results must be considered for error. Hopefully this tool will remain in use for future set-top box development and research.

2. Future Development

More Processes for More Realism

Though the simulation software as it stands appears complex, many smaller components of the actual TiVo operating system, kernel and hardware are not modeled yet in the simulation. Some components/processes to be modeled in the simulator include:

A. The loading and processing of TiVo user interface. Bursty HDD traffic to load flash and images needed to generate the UI.
B. Linux OS. The TiVo kernel is a private distribution of the Linux 2.6 kernel. The OS has connection, self-maintenance and processing overhead not currently modeled in the simulator.
C. Dual-HDD operation. The hardware team at TiVo has contemplated moving to a two HDD architecture. Adding similar functionality to the simulator would allow for TiVo engineers to maximize efficiency of such a system

A Post-Plotting GUI

Another simulation feature discussed with my superiors in multiple meetings was a post-processing graphing function. When the simulation completed, a few dialog windows would open up displaying graphs of the collected simulation data on the Y axes and simulation time on the X axes. The graphs would be color-coordinated to show errors and warning levels. A user could then cross check those sim times, where errors have occurred, on the X-axis against the

generated log messages with similar time stamps to find the source of the error.

One graph would display the used space in all the buffers over time. This graph would provide memory insight to engineers, pointing out time simulation time values where buffers overflowed or underflowed. Another graph would show the state of the HDD over the time of simulation. Different colors on the graph would correlate to different states such as reading, writing, flushing cache, ATR, RAM transport, etc. This graph could be used in accordance with the buffer graph for visual understanding of error messages.

VI. APPENDIX

1. References

http://www.python.org/
"Python Programming Language – Official Website." Python Programming Language – Official Website. N.p., n.d. Web. 11 June 2013.

http://simpy.sourceforge.net/
"SimPy Simulation Package." SimPy Home. N.p., n.d. Web. 11 June 2013.

http://www.riverbankcomputing.com/software/pyqt/intro
"What Is PyQt?" Riverbank. N.p., n.d. Web. 11 June 2013.

http://www.riverbankcomputing.com/software/sip/intro
"What Is SIP?" Riverbank. N.p., n.d. Web. 11 June 2013.

http://www.eclipse.org/org/
"About Eclipse." About the Eclipse Foundation. N.p., n.d. Web. 11 June 2013.

http://pydev.org/
"What Is PyDev?" PyDev. N.p., n.d. Web. 11 June 2013.

2. Acknowledgements

TiVo Corp.

Thank you to my supervisors and idea-generators at TiVo: Mukesh Patel, Sudhakar Rao and David Platt. Also, this project could not have been completed without the set-top box system knowledge of other TiVo firmware/hardware engineers: Srinivas Jandhyala, David Harrison and Kurt Heaton.

Cal Poly San Luis Obispo

To all the teachers their hard work and staff for the professional connections with TiVo. Finally, I would like to thank Dr. Christopher Lupo for advising the conversion from industrial tool to computer engineering senior project.