Low Cost, Transportable Power Platform and Flexible Interface for Multiple Hardware Protocols

Adrian Torres, Pradeep Batra, Brian Tsang, and Tsunwai Gary Yip
Rambus Inc. Sunnyvale, California USA
www.rambus.com

***Abstract* — this paper presents a flexible, portable hardware platform that allows software to control a DUT's environment accurately, inexpensively, and quickly. It significantly shortens test time allowing products to be introduced into the market faster and cheaper. Though the entire characterization environment is beyond the scope of this paper, three major hardware components of the environment are presented. They are the Labstation interface module (LIM), the Labstation power board, and the Labstation power module.**

The LIM is a flexible interface which serves as the hardware link between the host controller and the system-under-test. It translates USB packets sent from the host to commands compliant with any number of protocols including JTAG, SPI, I^2C, GPIO and other customer specific protocols. Thus, the LIM provides the links to access low-level registers of a test chip and manage the peripheral components of the test system. The power board contains twelve slots for power modules, an FPGA, and a variety of clock ICs to support the DUT. This modular architecture allows for ease of upgrade and re-design of the power modules to meet the specific needs of future DUTs. Power modules are designed to provide accurate power while using common off-the-shelf parts. It can output a wide voltage range while accurately measuring current during testing.

I. Motivation

The market requirements and specifications of an electronic system drive the development of a prototype that turns the initial concept of the system into a revenue-bearing final product. When the product happens to be an ASIC or silicon technology, the prototype often serves multiple purposes during the various phases of development and productization, such as initial bring up, characterization of component and system, product demo and launch. It is not efficient to build different versions of the system for each of the phases. Instead, re-using as much of the hardware and software as possible can shorten the time-to-market. This strategy has been proven successful in the development of memory technology for game consoles and HDTV using the LabStation environment [1, 2].

A brief overview of the LabStation environment will be presented. The focus of this paper is two re-configurable hardware sections of the LabStation environment, which are namely,

1. architecture and operation of the LabStation Interface Module (LIM), and
2. architecture of the power platform

II. Overview of Hardware Platform

Figure 1 shows the key hardware components in a typical LabStation environment. From the host computer, commands for control and testing are initiated and sent through the USB port to the command translator called the LabStation Interface Module (LIM). The LIM translates information encoded in the USB packets to the appropriate protocol for a target device on the power platform or test platform.

The power platform provides a base set of re-configurable resources that can support different phases of development. These resources include programmable power sources, high-speed clock generators, voltage references and an FPGA for application-specific test features. They generate the power rails for the test platform, which contains the prototype and the supporting peripheral devices required for proper operation or testing.

The architecture of the hardware in the LabStation environment is designed to be modular, thus allowing the power and test platforms to scale with the complexity of the prototype under test at a low cost. The modular nature also allows for easy transport of the hardware platform to different locations, such as customer sites or conferences.

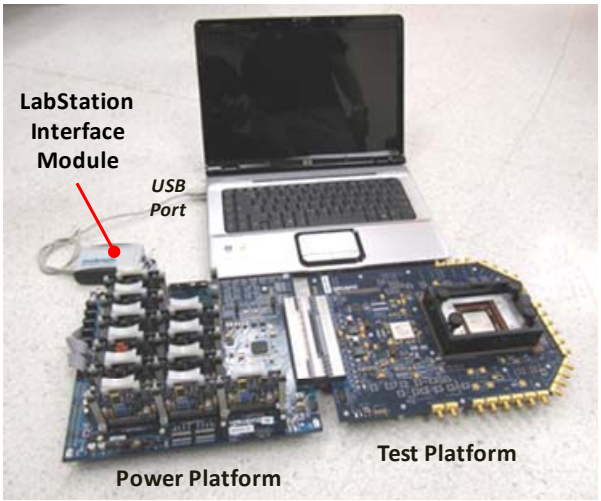


Figure 1: Hardware in a LabStation Environment.

III. LabStation Interface Module (LIM)

An important component of the LabStation environment is the LabStation Interface Module (LIM). Its key components are shown in Figure 2. The LIM serves as the conduit for communicating commands and transfer of data between the host PC and the test platform.

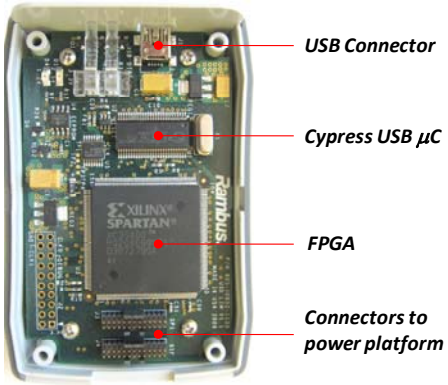


Figure 2: Key components in a LabStation Interface Module.

The architecture of the LIM and its operation are illustrated graphically in Figure 3. The USB packets being sent to the platform are received by an off-the-shelf USB microcontroller, which translates the USB protocol to a generic parallel interface. The interface is connected directly to an FPGA.

The LabStation software operates on the host PC and manages the commands and data directed to the test platform. The software takes operations from the LabStation GUI and/or scripts, and then generates USB packets with the corresponding commands embedded in a LabStation packet format. The LabStation packet format is one aspect of the LIM architecture.

Within the FPGA, the microcontroller interface block steers the LabStation packets into the appropriate FIFO queue. There is a set of FIFO queues for each of the communication ports. Each port consists of a Multi-Protocol Controller and a set of Protocol Engines. The combination of the Multi-Protocol Controller and a Protocol Engine executes the LabStation command in the communication protocol specific to the target device on the intended platform.

The LIM architecture strikes a balance between the software and hardware features. For example, the software handles errors during the transfer of LabStation packets while the hardware handles the electrical timing and signaling. This is an effective approach since the software application can implement the complicated error handling algorithms and keep track of the packets that may need to be re-transmitted. The complicated (conditional) logic is implemented in the LabStation software application and thus frees up FPGA resources to allow for more protocol engines or other features if required.

From a high-level architectural perspective, the two primary functions of the LIM are namely,

1. handle data transfer over using the USB protocol, and
2. translate commands to the proper protocol for a given platform device.

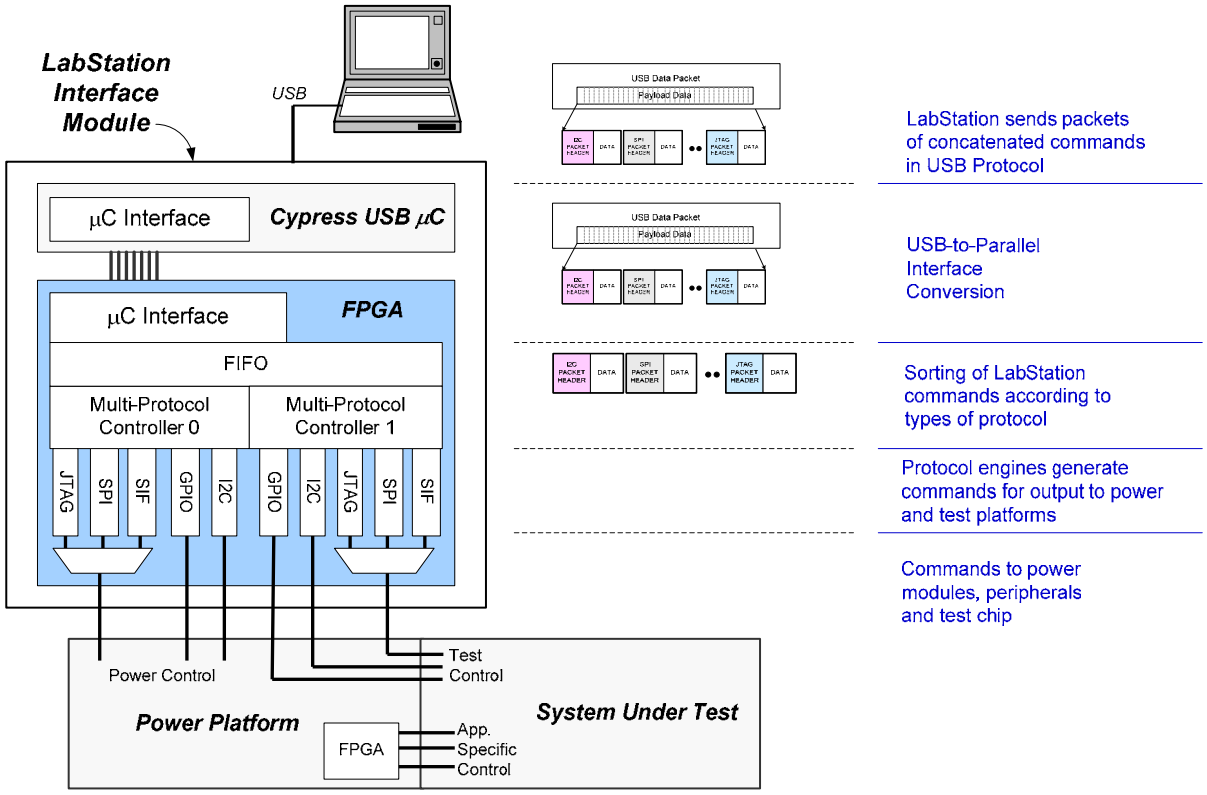


Figure 3: Architecture of LabStation Interface Module (left) and conversion of LabStation commands to protocol specific commands (right)

A. LabStaion Packets

As shown in Figure 4, a USB data packet contains multiple LabStation packets in its payload. The LabStation software concatenates packets to utilize the USB bandwidth for maximum throughput.

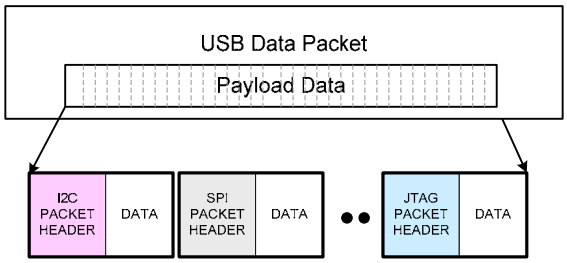


Figure 4: LabStation Packet Concatenation

Each LabStation packet consists of a header and a data section. The header consists of architected fields which are used to encode the control information for the FPGA logic. The header has fields to determine the length of the data section and how the data is processed in the transaction. For example, a write operation passes the data to a device on the platform. In cases where LabStation software is reading data from a device, the packet header is not required. This is due to the fact that LabStation software is the master for data transfer and is aware that data is available in the FIFO queue.

The header includes fields to encode which protocol engine to employ and may invoke special operations that the engine offers. For example, the timing of the clock signal may be adjusted before a transaction begins. There is a wide variety of encodings that have been architected into the header format. There are sufficient encodings reserved for additional features or to support additional protocols.

In summary, the packet header has been architected to be scalable. The same header format is used to encode a variety of protocol commands and FPGA logic operations.

B. Microcontroller

The design uses a low-power USB microcontroller from Cypress Semiconductor. The microcontroller includes an 8051 microcontroller, a USB PHY and a Serial Interface Engine which handles the USB protocol. The 8051 microcontroller operates according to the firmware instructions stored in an I^2C EEPROM. The primary role of the microcontroller is to transfer data between the USB interface and the parallel interface which is connected to the FPGA.

The microcontroller also controls a clock buffer which can be programmed to generate a range of clock frequencies on independent outputs. The outputs of the clock buffer feed the FPGA logic. This enables different portions of the FPGA logic to operate at different data rates. This is illustrated further in the next section.

C. FPGA

The FPGA design consists of four functional blocks: the microcontroller interface, the FIFO queues, the Multi-Protocol controller and the protocol engine. This modular approach allows the LIM to easily scale to support more communication ports or engines. A communication port consists of a Multi-Protocol controller with protocol engines.

1) Microcontroller Interface

The Microcontroller Interface logic block shown in Figure 3 has two primary functions. First, the logic is designed to perform the communication to the USB microcontroller over a parallel interface. Second, it performs the first level of decoding to steer commands and data to the downstream logic in the FPGA.

The FPGA logic also contains control and status registers. As such there is logic in the interface block that determines if a command is meant for the FPGA registers or the platform. The commands meant for the FPGA registers are handled directly by the interface logic. The commands that target the platform are steered to the FIFO queues by the interface logic in the form of LabStation packets.

2) FIFO Queues

There is a dedicated set of FIFO queues for each of the Multi-Protocol Controllers as shown in Figure 5.

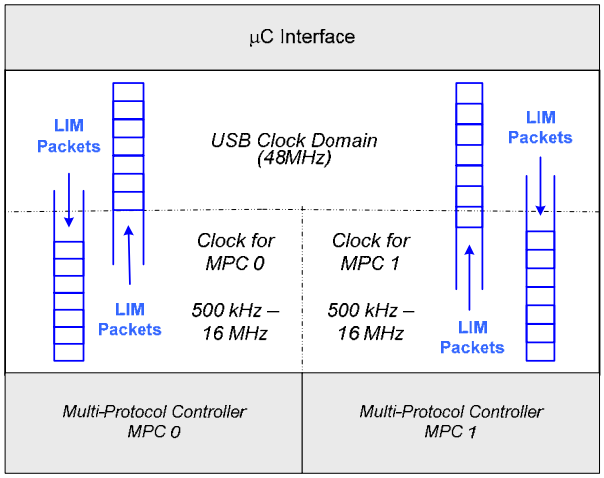


Figure 5: Clock Domain Boundaries

The FIFO queues are employed to cross clock domain boundaries between the microcontroller interface and the communication ports. All of the FIFO queues interface with the microcontroller interface block in a common 48MHz clock domain. However, there is an independent clock domain for each paired set consisting of FIFO queues and a Multi-Protocol Controller. This architectural feature allows the underlying logic to operate at distinct frequencies per the requirements of the test platform. For example, the platform may require a JTAG interface operating at 16MHz and separate SPI interface operating at 8MHz. The LIM communication port frequencies are easily configured via the LabStation software.

3) Multi-Protocol Controller (MPC)

The Multi-Protocol Controller is a state machine that monitors the FIFO queues for incoming LabStation packets. When a packet is detected, the MPC decodes the LabStation packet header to determine the appropriate action. If the header specifies that there is data to be processed, then the port

controller employs the appropriate protocol engine to process the data. Otherwise, the header may dictate a special operation such as temporarily overriding the GPIO settings or initiating a specific reset sequence. In some cases, the controller may operate two protocol engines in parallel.

The Multi-Protocol Controller is the block that is modified to introduce a new protocol engine in the LIM.

4) *Protocol Engine*

As indicated above, the Protocol Engine is mastered by the Multi-Protocol Controller. The primary role of the engine is to follow the given protocol standard to transmit or receive data. Each Protocol Engine is designed to handle a specific protocol such as JTAG, SPI or I^2C. The engine implements the timing characteristics of the protocol and instantiates the appropriate FPGA IO buffers required to interface to the devices on the platform.

IV. LIM Operation and Usage Model

The LIM is effectively a generic device that may be used with any hardware platform. It is easily configurable to communicate to devices using a variety of standard protocols as well as custom, proprietary protocols. Based on the architecture, however, it is a slave device to the LabStation software. Therefore, the LabStation software must be used to control the LIM.

Although there are many ways to interface the LIM to hardware platforms, Figure 3 illustrates an effective, scalable approach. The illustration shows one port (MPC-0 and associated protocol engines) connects to the power platform and a second port connects to the system under test.

Each port is independent of one another; therefore the ports are configured based on the platform requirements. The first port is dedicated to the power platform and controls all of the associated devices. The second port is dedicated to the system under test. In most applications, the power platform provides the resources required by the system under test such as power, clocks, voltage references, etc. In some cases additional features are required by the system under test. In such cases, additional features that must be controlled by software can be implemented directly on the system under test. The second port can then control the associated devices using the required protocols and operating at the specified frequency, independent of the first port's configuration. Therefore, the power platform interface does not interfere with the system under test interface.

V. Power Platform

The power platform shown in Figure 6 provides the logic and infrastructure to test and debug a system. With 12 module connectors, an FPGA, six DACs, six ADCs, and four clock chips; the power platform is able to supply a data bus, power, reference voltages, and an assortment of clocks and GPIO pins. Table **1** is lists out the features of the board. The power platform provides these features to the system under test through an impedance-controlled, high pin count VHDM connector.

To handle the potentially large amount of power, the platform has five 12V inputs for external AC/DC bricks to connect. The different voltage rails generated by the platform run through a VHDM connector to supply up to 80A of programmable power to the test platform.

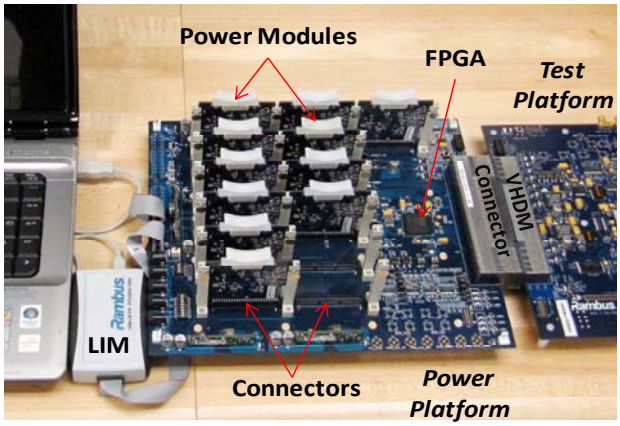


Figure 6: Power platform between LIM and test platform.

Table 1 Features of Power Platform

1	Up to 12 power rails
2	6 programmable high speed clocks
3	Support for I^2C, JTAG, SPI devices
4	24 GPIO pins
5	60 pin parallel data bus

A. *Module Connectors*

The majority of the power platform is taken up by 12 module connectors. Each connector has three supplies, 3.3V, 5V and 12V, to support the programmable power module installed in it (see Figure 6). The connector also has an SPI and I^2C ports to provide logic support to any IC on the module.

To maximize space, facilitate cooling of components, and allow flexibility in the size of future modules, the connectors are placed vertically allowing modules to stand perpendicular to the power platform. A plastic guide is used to facilitate the insertion and removal of a module and to protect the connector when in use. It allows modules to be locked for additional stability during transit.

B. *Power distribution*

The design of the power distribution was specifically meant to be flexible and allows users to configure the power platform to meet the needs of the system under test. The distribution is illustrated in Figure 7. The source of all the power is the 12V rail which is provided externally by power bricks. In most cases, the power consumption of the power platform is less than 50W and a single 50W AC/DC adapter is sufficient to support the testing. This enables easy portability during the typical low power usages. However, in cases where additional

power is needed, up to five adapters can be connected in parallel to provide the platform up to 250W of power. In the extreme case where all DC/DC converters are fully loaded there is an option to connect an ATX power supply.

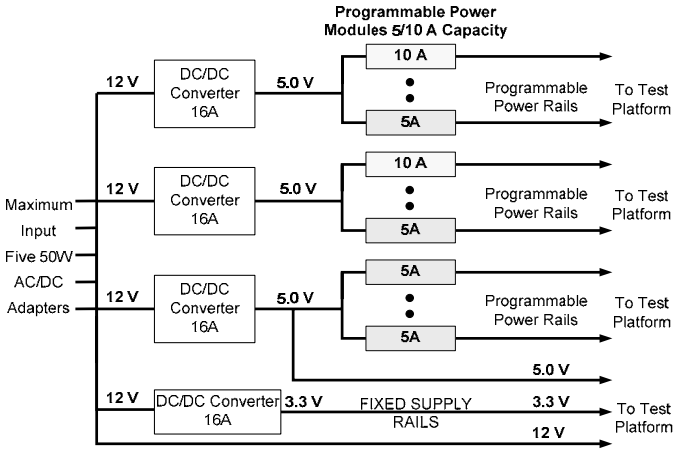


Figure 7: Generation and distribution voltage rails on the power platform.

There are four DC/DC converters that can each distribute 5.0 V power to four or less programmable power modules based on the needs. The power limit of each converter is 80W which can be distributed depending on the needs of the system. For example, a user can choose to populate only two out of the four modules so that each module could consume 30W.

The power platform was designed so that 8 modules can provide 5 Amps of current and 4 modules can provide 10 Amps of current. The power is supplied to the test platform through specific low resistance, high current pins of the VHDM connector and special care was taken to prevent neck down of power planes and single via layer transitions.

There are also three fixed 3.3V, 5V and 12V voltage rails to supply power to the test platform.

C. FPGA

The FPGA is used to provide logic for the control of both the DUT and the power boards. Through a register interface which is graphically presented in the Labstation software, users can easily turn on/off power, control clocks, or set specific GPIO nets that are routed to the DUT. There are three 8-bit GPIO ports routed through the VHDM. One port uses a standard 3.3V CMOS IO. The two other ports have programmable voltage levels allowing easy hook up to a variety of different IO standards.

Though the FPGA is generally controlled through software, user defined push buttons and dip switches are also available. The FPGA constantly monitors the state of the modules and is able to detect failures in any module. A bank of LEDs allows the FPGA to communicate any module failures or errors from the FPGA state machines to the user.

In addition to the three 8-bit GPIO ports, the FPGA contains a fast, parallel, source-synchronous bus to pipe data from the FPGA to test chips. The 60 bit wide data bus features a programmable voltage swing, the ability to de-skew the clock respect to the data, and has an aggregate bandwidth of 2.4 GB/s.

To control the large number of components on the power and DUT boards, the FPGA instantiates a large 6-to-64 decoder for the chip selects of devices and acts as the buffer to the data in and data out pins.

D. Clocks

Twelve adjustable, differential, high speed clocks are generated on the power platform along with six slower, single ended clocks from 2 separate crystals. The topography is shown in Figure 8. The six low speed clocks all have a 3.3V swing and are comprise of two 27 MHz clocks, two 33 MHz clocks and two 48 MHz clocks. The high speed clocks can be finely adjusted from 100 to 600 MHz with an adjustable swing of up to 2.5 Vpp. All clocks have spread spectrum control and can individually be enabled through the FPGA or dip switches.

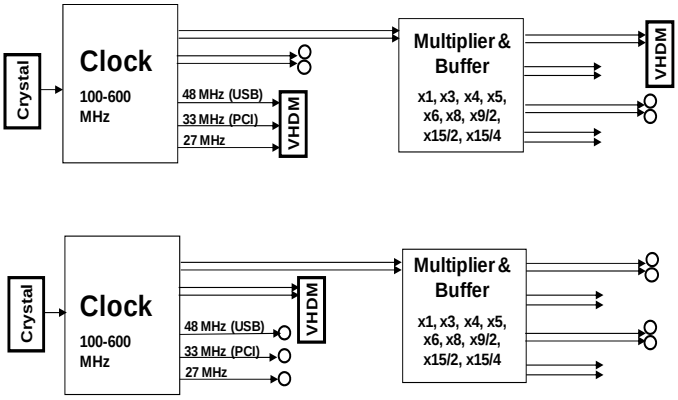


Figure 8: Clock topography on the test platform.

Three of the single ended clocks are routed through the VHDM connector to the test platform. The other three are routed to SMA connectors and can be connected to external equipment or the test board through cables. Of the twelve high speed clocks, two pairs are routed through the VHDM to the test platform and four are routed to SMA connectors.

Though the generated clocks are not all on the same clock domain, there are enough clocks in each clock domain to meet the needs of most test requirements.

E. Voltage Reference

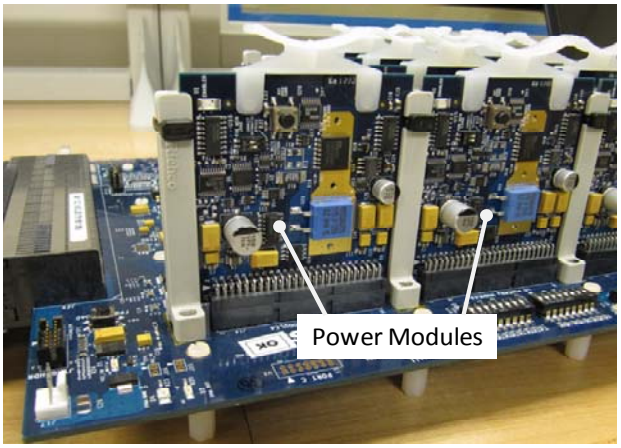
The power platform supplies six reference voltages along with a corresponding ability to measure the voltage at the load. Though not as precise as a band-gap reference, the reference voltage can be programmed up to 3.3V with a maximum current drive of 20mA.

By measuring the voltage at the point-of-load, the power platform allows users to read and adjust the voltage output until

the optimal voltage setting is reached at the load. The accuracy and adjustability of the reference voltage is under 1 mV.

VI. Power Modules

Though the power platform was designed to control a variety of different modules, the power module is essential for normal operations. Its functions not only include providing regulated power, but also measuring the voltage at the point of load, adjusting the voltage, and accurately measuring the current of the load over a wide range. The module also contains non-volatile memory to store values which compensate for the errors in the components and increases the accuracy of the module.



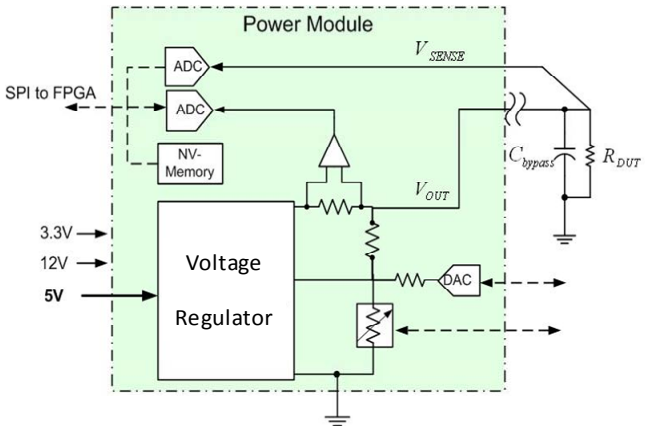


Figure 9: Programmable power module.

Figure 9 shows the high level block diagram of the power module and the main features that it contains. The output voltage can be adjusted 2 ways. The digital pot allows the power up voltage to be programmed by adjusting the feedback resistor of the regulator. The full range of the potentiometer is able to change the output from 750mV to 3.3V. The feedback resistor is for coarse adjustments of the output voltage. For fine adjustments, a DAC is used to inject or remove current from the feedback network and is able to change the output voltage by ±250mV.

The voltage at the point of load is measured through Vsense which comes from the test platform and is routed through the VHDM and module connector. This feedback path ensures that the voltage seen by DUT on the test platform is accurate and controllable.

A shunt resistor is used to measure the current supplied to the DUT. Because the shunt is placed before the feedback network (see Figure 9), the regulator automatically adjusts Vout to be constant independent of current draw and voltage drop across the shunt. However, because of this, the feedback resistors of the regulator must be sized large enough such that the current through the feedback network is insignificant compared to the current draw from the DUT to ensure accurate current measurements. The value of the shunt resistor is directly related to the expected current draw of the DUT. In fact, depending on the current range that needs to be measured, the shunt resistor value may change.

The non-volatile memory on each module stores values to compensate for variations and inaccuracies of the components of the module. At initial bring up each module is connected to lab equipment and a known load board. Voltage and current measurements are taken by the module and compared with data from the lab equipment. By comparing the two results and storing the error term, a user can later use that value to reduce the offsets and errors of each module.

A. Specifications

Table 2 Power Module Specification

$Vout_{MAX-A}$	3.3V*
$Vout_{MIN-A}$	750mV*
$Vout_{MAX-B}$	650mV*
$Vout_{MIN-B}$	0mV*
I_{MAX}	5A
I_{MIN}	50mA
$V_{ACCURACY}$	±5mV
$I_{ACCURACY}$	10%

Table **2** lists out the capabilities of the programmable power module. For most applications, the module is configured to output 750mV to 3.3V. However, for some low power applications, a voltage rail less than 700mV is needed. In these cases, the module's feedback topography can be reconfigured through software such that the output supplies 0 to 650mV. As we will see in the next section, the $I_{ACCURACY}$ of the module is guaranteed when the voltage of the output is less than 2.5V and the current is greater than 50mA.

B. Performance

The design of the power module was measured by connecting the output of the module to an adjustable load board. By changing both the load and the output voltage, the

module was tested over a wide range of currents at different voltage points.

Figure 10: V_{OUT} measured in infinite persistence while adjusting the current drawn.

Because the shunt resistor was placed within the feedback loop of the regulator, varying loads may induce noise on the output voltage of the module. To test this, we measured the output voltage at the load while drastically adjusting the load and current draw. Nothing above the noise floor of the probe and scope was found as seen in Figure 10.

Figure 11 shows the linearity of the output voltage of the module when adjusting the DAC and digital pot. As expected V_{OUT}, is linear to the 12 bit DAC setting and consistently spans 500mV independent of the voltage level. The output saturates at 3.3V due to the limitations of the regulator.

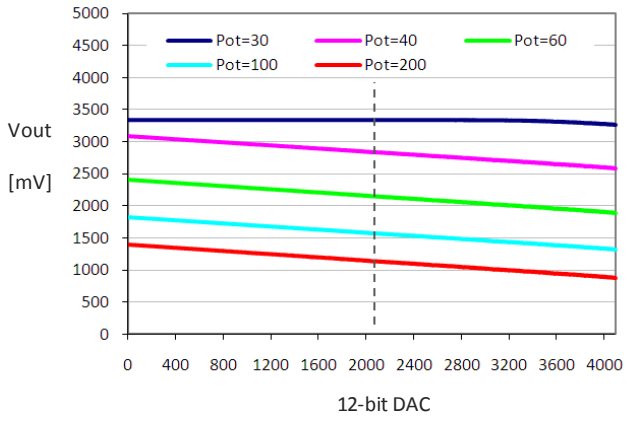


Figure 11: Linearity of V_{OUT} over the input range of the 12-bit DAC.

To measure the current accuracy, we connected a digital multi-meter in series with the power delivery and compared the measurements done by the module to the values given by the multi-meter. We adjusted both V_{OUT} and the load resistance to see the affect each had on the accuracy. Figure 12 shows the accuracy as a function of the Vout or current draw. The deterioration of the current measurement is mainly based on V_{OUT} and not the current draw. In most cases, the current sensing circuit is less than 4% off from the measurement taken with lab equipment. However, as V_{OUT} increases above 2V, the accuracy of the on board measurement deteriorates. When V_{OUT} is less than 2.5V, the error is less than 10%. However, at the spec limit of 3.3V, the error increases to 30%. Since the typical DUT needs less than 1.8V power, the performance of the measurement is excellent.

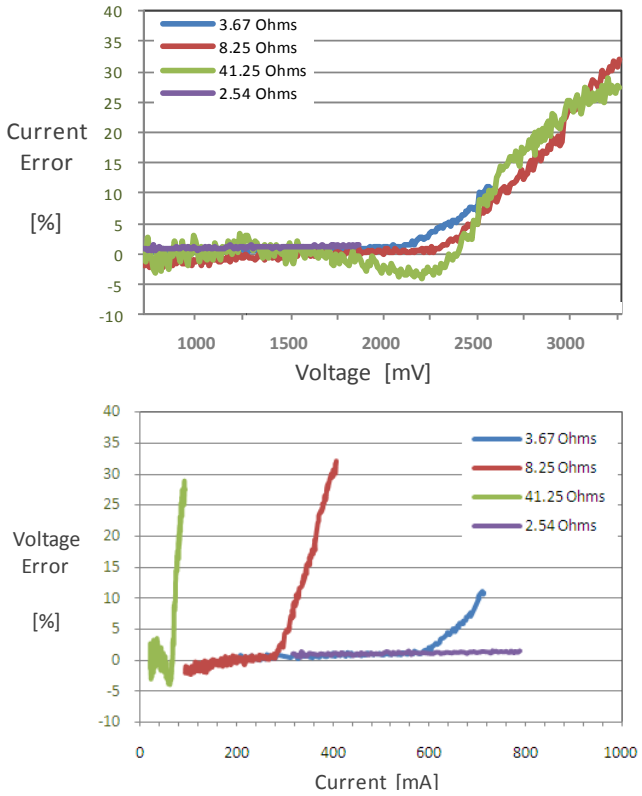


Figure 12: Errors in current and voltage for different resistance load.

VII. Conclusion

The architecture of the LabStation interface module, power board and programmable power modules provides a versatile and efficient infrastructure that can be re-deployed to support the characterization of different IO cell technologies. Their small form factors and the use of off-the-shelf components to build them offer the added advantages of low cost, and ease of transport. Time-to-market is therefore reduced through shortening the development and productization of a new system.

References

[1] P. Batra, A Torres, W. Ng, A. Vaidyanath, P. Yeung, and T. G. Yip, "LabStation: A Low Cost, Scalable, High Throughput Characterization Environment," Silicon Valley Test Conference, San Jose, California, USA, November 2010.

[2] P. Yeung, A. Torres, P. Batra, "Novel Test Infrastructure and Methodology Used for Accelerated Bring-Up and In-System Characterization of the Multi-Gigahertz Interfaces on the Cell Processor," date, pp.137, 2007 Design, Automation & Test in Europe Conference & Exhibition, 2007.

Authors Biography

Adrian Torres is currently a Senior Manager of the Product Engineering Team at Rambus. He led the effort to develop the LabStation Interface Module and the modular power platform for the LabStation environment. Upon joining Rambus in 2003, his primary role was to co-develop test and validation platforms for the high-performance interfaces, FlexIO processor bus and XDR memory bus, in the PlayStation 3 game console. He has also managed system teams in various Rambus advanced developments such as the recent Mobile XDR initiative. Presently he manages a team of product engineers to ensure Rambus PHY products are ready for full production. Prior to Rambus, he has 5 years of experience in testing, validation and in-system diagnostic tools development for Itanium based enterprise server at HP. He received his B.S. in Electrical Engineering and B.S. in Computer Engineering from the University of California at Davis.

Pradeep Batra is currently a Senior Principal Engineer and Software Architect at Rambus Inc. He is the inventor of the LS Script language for LabStation. His current research focus is operating systems for mobile platforms. Since joining Rambus in 1992 he has developed software tools for bring-up, characterization and diagnostic of RDRAM, XDR, and DDR memory subsystems. His tools were instrumental to the development of products including Nintendo64, RDRAM based PC's, graphics cards, Playstation3, and TI DLP projectors. Pradeep received his BS in electrical engineering from IIT Delhi and his Masters in Electrical Engineering from UCLA. He has authored several papers and has over ten issued patents. Prior to joining Rambus he was a member of the SPARC processor design group at Sun Microsystems.

Brian Tsang joined Rambus Inc. in May 2002. He is currently a Principal Engineer in the System Technology Team studying trends in mobile OS and advanced system technology. He was a key contributor in the board level design and characterization of the FlexIO processor bus for the PS3. For the LabStation, he led the design of the power board and programmable power module, and is responsible for architecting the software module for the power platform. Since joining Rambus, he has worked to develop software, test platforms, and tests for a variety of high performance interfaces. These included RDRAM, FlexIO processor bus, and PCI express. Brian graduated from Stanford University, where he received his BS in Computer Science, and BS and MS in Electrical Engineering.

Tsunwai Gary Yip joined Rambus Inc. in 2000 and is currently a Senior Principal Engineer. His interests are in the areas of EMI/EMC of mobile systems, low phase noise clocks for 10 - 30 GHz data links, phase noise analysis of PLLs, and thermal analysis of system and components. He developed many system clock solutions for LabStation test platforms, and the temperature control for maintaining test chips between -40°C and 125°C during characterization. He is also active in new technology trends in mobile handsets and platforms, consumer electronics and PC graphics. Dr. Yip is an Associate Fellow of the American Institute of Aeronautics and Astronautics. He was a recipient of the Ralph Teeter Award from the SAE International for outstanding young educator in aerospace technology and was recognized as a Recent Outstanding Alumnus by the University of Illinois, Urbana, where he received his Ph.D. and M.S. degrees.