Preprints of the 19th World Congress
The International Federation of Automatic Control
Cape Town, South Africa. August 24-29, 2014

A Practical, Hybrid Approach to Faster-Than Real-Time Power System Analysis and Control

Anthony S. Deese, Ph.D.*, C. O. Nwankpa, Ph.D., Stephen Coppi*, Tim Nugent***

** The College of New Jersey, ECE Department, Ewing, NJ 08628 USA (tel: 607-771-2779; e-mail: deesea@tcnj.edu).*
***Drexel University, Philadelphia, PA 19104 USA (e-mail: con22@drexel.edu)*

Abstract: This paper examines how custom computing hardware for fast power system analysis may be adapted to improve the user experience as well as its commercial-viability. One prime example of such custom hardware is the FPAA-based analog emulator developed previously by the authors. The authors propose development of a USB-based actuation and data acquisition interface that allows analog computational tools to work seamlessly with one or more commercially-available power system analysis software packages. A commercially-available software application will be tasked with user interaction and post-processing, while custom hardware is tasked with faster-than real-time power system analysis.

Keywords: power system analysis, analog emulation, actuation and data acquisition, USB

I. INTRODUCTION

This work builds upon previous research in analog emulation of power system dynamics via field-programmable analog array (FPAA) technology [2009a, 2011b, 2011a, 2007, 2013]. The research has generated multiple electronic analog computer prototypes and peer-reviewed publications [2009a]. In this paper, the authors discuss how existing analog emulator prototypes (like those shown in Fig. 1) may be adapted to improve the user experience as well as its commercial-viability. They propose development of a USB-based actuation and data acquisition interface that allows analog computational tools to work seamlessly with one or more commercially-available power system analysis software packages (e.g. PSCAD). An overview of this technique is presented in Fig. 3. The software facilitates user interaction and post-processing, while analog hardware provides faster-than real-time analysis capability.

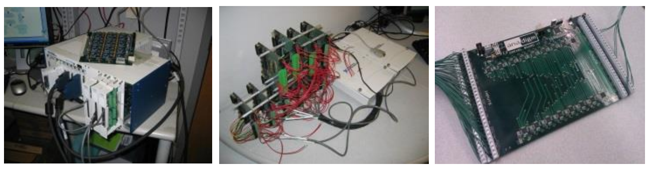

Fig. 1. First (Left), Second (Middle), and Third (Right) Analog Power System Emulator Prototypes Referred to in this Paper

II. BACKGROUND

In engineering and the sciences, digital computers are often utilized to predict the behavior of a non-linear system via simulation, a software implementation of iterative numerical techniques like the Newton-Raphson [2012a, 2011a]. This technology, due to advances in personal computing, is easy to operate and capable of producing precise results. Its speed, in comparison to hand calculations, allows the practical implementation of numerical methods that previously had none [2009a]. However, one limitation of digital simulation is that length of time required to simulate a solution is dependent on the number and complexity of nonlinear expressions [2009a]. The complexity of a power system simulation grows with third power of system size [2012a].

For the same purpose, analog computers use emulation. In it, a non-linear system model is implemented as a set of reconfigurable analog circuits referred to as the emulator [2011a]. This hardware is then actuated, initialized, and allowed to settle to a constrained solution. The user acquires results through observation of this hardware via voltage and current measurement devices. One may ask: Is it worth the additional effort associated with hardware design and construction to perform such analyses via analog emulation that may be performed digitally? The authors have shown in previous works that the answer is "yes." Because analog emulation abandons the use of iterative numerical techniques, the length of time required to yield a solution (this excludes the effects of actuation and data acquisition) is fully controllable and independent of the dimension of the system model [2012a]. Analog computation has potential to perform many types of nonlinear analyses significantly faster than is possible digitally; however, digital simulation is still generally utilized over analog alternatives for such studies. Several reasons exists, including potential nonlinear behavior, high power consumption, limited precision, lack of reconfigurability, and size of analog technology. Given this, what motivation exists to study analog emulation as an alternative to digital simulation for nonlinear system analysis? One prime example is the emergence of new analog tools like the field-programmable analog array (FPAA).

The field-programmable analog array (FPAA) represents a new frontier in analog technology, one which benefits from the application of large scale integration (LSI) methods to a reconfigurable, switched capacitor design. It is essentially a blank canvas for the development of custom analog hardware, composed of configurable analog blocks (CAB) which may be actuated and interconnected digitally. The most basic definition of the FPAA's operation is a configuration file which, when uploaded to hardware, configures and interconnects all CAB's. The method in which this configuration is generated depends on model and manufacturer. Fig. 2 demonstrates the flow of information required to configure a single FPAA chip. It is similar to that of the FPGA; however, logic blocks would take the place of CAB's.

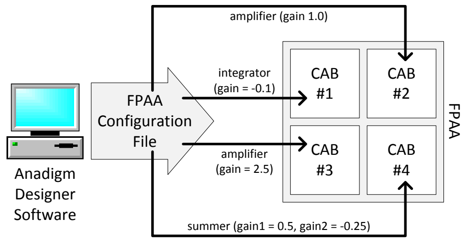

Fig. 2. Demonstration of Configuration File Upload to FPAA Hardware

Although some FPGA's provide limited analog capability, it is most often implemented through utilization of high-performance analog-to-digital and digital-to-analog converters. Such hardware does not provide the full capability of an analog circuit and cannot be successfully employed for analog emulation.

III. MOTIVATION

The authors realize that users, particularly in industry, prefer tools similar to those they have used previously. As such, the authors of this work employ a solution that incorporates commercially-available software and hides any evidence of the tool's experimental nature from the user, providing an experience identical to that of applications like Siemens' PSS/E, Mathworks SimPower, Manitoba HVDC PSCAD, and PSerc Matpower. The prototype will utilize a digital computer (PC) running one of the aforementioned applications to facilitate user interaction but rely on an FPAA-based analog hardware accelerator to improve computation speed and overall performance.

The work discussed in this paper exploits the complementary strengths and weaknesses of digital simulation and analog emulation, as applied to nonlinear system analysis and optimization. In grossly simplified terms, it may be described as follows. Digital simulation is generally precise but slow; analog emulation exhibits opposite characteristics. It is for this reason that the authors employ a mixed-signal computing solution, one that employs digital as well as analog devices and draws upon the strengths of both. One of the most innovative aspects of this work is the utilization of FPAA technology in the place of traditional analog components.

IV. PROBLEM STATEMENT

Although a number of published works examine the application of analog and FPAA-based computation technologies to load flow analysis, adoption by industry and academia is limited. How can researchers adapt existing analog emulator prototypes (like that shown in Fig. 1) to improve the user experience as well as its commercial-viability?

V. PROPOSED SOLUTION

The authors propose development of a USB-based actuation and data acquisition interface that allows analog computation tools to work seamlessly with one or more commercially-available power system analysis software packages (e.g. PSCAD). Software facilitates user interaction and post-processing, while analog hardware provides faster-than real-time analysis capability. This paper focuses predominantly on static load flow analyses.

VI. METHOD OVERVIEW

6.1 System Components

Fig. 3 provides an overview of the proposed mixed-signal computing tool. It is composed of three primary pieces:

- *Power System Simulation Tool* – a commercially-available software application that allows users to perform a variety of power system studies including stability, contingency, and optimal power flow.
- *Analog Power System Emulator* – is an analog computational tool designed to mimic the behavior of a power system and, in turn, perform various static as well as dynamic power system analyses. It may be actuated, initialized, and allowed to settle to a constrained solution. Results are acquired through observation of the voltages and currents it generates. Several examples are shown in Fig. 1.
- *USB-Based DAQ System* – is hardware designed to link the simulation and emulation tools, performing analog hardware actuation and data acquisition as needed.

6.2 Management Algorithm

For this proposed solution, the user interacts with the power system simulation tool normally. A graphical user interface allows she / he to define the parameters of the study ($\underline{u}$) as well as acquire simulated results (y_{sim}). Normally, a *management* algorithm embedded within the simulation tool will perform several tasks:

- *Interpret Study Parameters* – The management algorithm interprets study parameters ($\underline{u}$) and generates the configurations ($\underline{u_i}$) associated with individual load flow analyses. For example, a single-contingency study must be broken down to multiple power system configurations, each corresponding to the failure of a single component. Note that $\underline{u}_i$ represents the input/configuration employed by an i^{th} load flow analysis.

- *Define Initial Conditions* – The management algorithm generates an initial condition ($\underline{x}_{0,i}$) for each load flow analysis. Often a *flat start* is employed. Note that $\underline{x}_{0,i}$ represents the initial condition employed by an i^{th} load flow analysis.
- *Interpret Load Flow Solutions* – The management algorithm interprets load flow solutions ($\underline{y}_{sim,i}$) and generates the comprehensive study result ($\underline{\underline{y}}_{sim}$) associated with $\underline{u}$. For example, the results of a single-contingency analysis must be extracted from multiple load flow solutions. Those that violate nominal operating limits are highlighted to the user. Note that $\underline{y}_{sim,i}$ represents the simulated solution of an i^{th} load flow analysis, as defined by the configuration $\underline{u}_i$ and initial condition $\underline{x}_{0,i}$.

Each variable in this paper is underlined according to its dimension. For example, the two-dimensional $\underline{\underline{y}}_{sim}$ matrix is composed of multiple one-dimensional $\underline{y}_{sim,i}$ arrays.

$$\underline{\underline{y}}_{sim} = \{\, \underline{y}_{sim,1} \quad \underline{y}_{sim,2} \quad \cdots \,\} \tag{1}$$

In the authors' proposed method, the management algorithm performs an additional task. It utilizes external analog emulation hardware to pre-process load flow results. For an i^{th} load flow analysis, the management algorithm supplies an appropriate configuration $\underline{u}_i$ to the analog emulator, acquires the emulated load flow solution $\underline{y}_{emu,i}$ it generates, and utilizes this solution to generate a better estimate of $\underline{x}_{0,i}$.

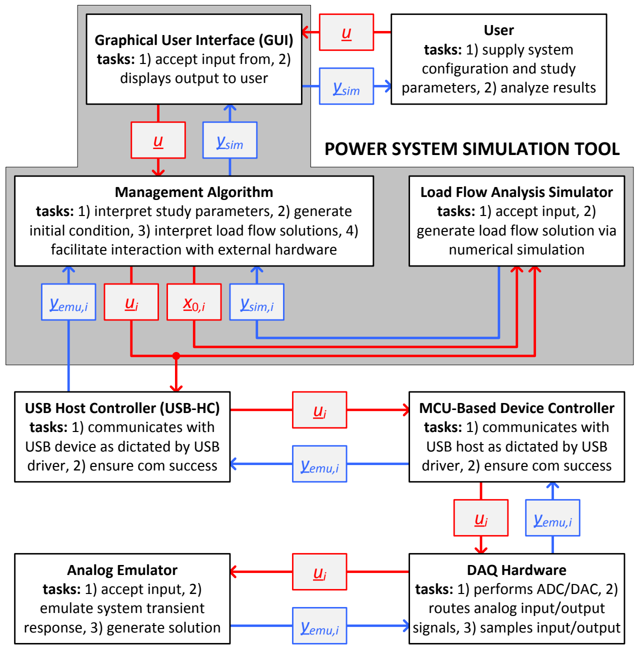
Graphical User Interface (GUI)
tasks: 1) accept input from, 2) displays output to user
u
y_sim
User
tasks: 1) supply system configuration and study parameters, 2) analyze results
u
y_sim
POWER SYSTEM SIMULATION TOOL
Management Algorithm
tasks: 1) interpret study parameters, 2) generate initial condition, 3) interpret load flow solutions, 4) facilitate interaction with external hardware
Load Flow Analysis Simulator
tasks: 1) accept input, 2) generate load flow solution via numerical simulation
y_emu,i
u_i
x_0,i
y_sim,i
USB Host Controller (USB-HC)
tasks: 1) communicates with USB device as dictated by USB driver, 2) ensure com success
u_i
y_emu,i
MCU-Based Device Controller
tasks: 1) communicates with USB host as dictated by USB driver, 2) ensure com success
u_i
y_emu,i
Analog Emulator
tasks: 1) accept input, 2) emulate system transient response, 3) generate solution
u_i
y_emu,i
DAQ Hardware
tasks: 1) performs ADC/DAC, 2) routes analog input/output signals, 3) samples input/output

Fig. 3. Overview of Proposed Hybrid Computing Solution

For hardware actuation, the information contained within $\underline{u}_i$ is routed to the PC's USB Host Controller (USB-HC) [2001, 2009b, 2012b] The USB-HC communicates with a USB Device Controller (USB-DC) embedded within the external hardware and transmits $\underline{u}_i$ along a standard USB message pipe. These controllers work together to ensure that no data is changed or lost. The device controller interfaces directly with DAQ hardware that generates and routes analog control signals to the emulator as defined by $\underline{u}_i$.

Once emulation is complete, the management algorithm requests that data acquisition begin. It instructs DAQ hardware to observe, perform A/D conversion on, and store the analog output signals that compose the emulated load flow solution ($\underline{y}_{emu,i}$).

VII. DAQ HARDWARE

USB 2.0 communication will be utilized to actuate as well as acquire data from analog emulation hardware. This custom DAQ hardware will be classified as a class FFh device, indicating that vendor-defined drivers are required for proper operation. USB 2.0 hardware allows transfer of data with a maximum rate of 480MBit per second as well as provides devices with a 5.0V, 1.5A-rated power supply. These rating dictate the speed of the actuation and data acquisition hardware as explained below [2001, 2009b, 2012b].

7.1 Actuation

For hardware actuation, configuration data is transmitted along a USB message pipe. Unlike faster stream-based alternatives, it facilitates bi-directional communication between host and device for acknowledgement of data transfer well as error correction. Information is transmitted as a set of USB transactions, each consisting of one or more packets [2001, 2009b, 2012b]:

- *Token Packet* (32-*bit*) – defines the direction of data flow, type of data to be transmitted, and destination of all subsequent packets. It is composed of six fields: 1) the 8-bit SYNC field is used to synchronize the clock of the device to that of the host, 2) the 8-bit PID field identifies the type of packet being transmitted, 3) the 7-bit ADDR field provides address of intended packet recipient device, 4) the 4-bit ENDP field facilitates routing of a packet within the recipient device, 5) the 5-bit CRC or Cyclic Redundancy Check field ensures proper data transmission, and 6) the 3-bit EOP or End of Packet field indicates that a packet is complete.
- *Data Packet* (*variable*) – is capable of transmitting up to 1024 bytes of data per packet within its DATA field. Like the token packet, it also contains SYNC, PID, CRC, and EOP fields.
- *Handshake Packet* (< 32-*bit*) – completes a transaction and ensures proper transmission of data. It is composed of SYNC, PID, and EOP fields.

The three-stage process shown in Fig. 4 is employed to transmit via USB the information required for actuation of a single variable within the emulator. It is referred to as a

3s1vUTAc (three-stage, single-variable, USB-based transmission for actuation) process. Each process communicated several pieces of information including [2001, 2012b]:

- MODE – This 4-bit data field defines the current mode of the DAQ system and analog emulator hardware.
  - MODE = 0 – denotes that the DAQ system is performing actuation. All actuation-related hardware should be active. Meanwhile, the emulator should accept new as well as maintain existing user-define initial condition values. Transition from this mode to MODE = 1 occurs once N_{In} analog input signals are successfully applied and transmitted.
  - MODE = 1 – denotes that actuation is complete and analog emulation has begun. All actuation-related hardware should be inactive, except for the sample-and-hold circuits that generate required analog control signals. The analog emulator should release its user-defined initial condition and monitor for solution convergence. Transition from this mode occurs once the analog emulator achieves steady-state.
  - MODE = 2 – denotes that analog emulation is complete and data acquisition has begun. All data acquisition-related hardware should be active. Meanwhile, the emulator should maintain the steady-state solution to which it has settled. Transition from this mode to MODE = 0 occurs once N_{Out} analog input signals are successfully acquired and transmitted.
- varNAME – This 20-bit data field defines the name of a control variable to be actuated as well as, in turn, the destination of an analog control signal within the emulator itself. This field allows the user to address more than 1 million distinct variables.
- varDATA – This 12-bit data field defines the value of a control variable.

The setup stage is composed of three packets, the second of which provides the DAQ hardware with MODE as well as varNAME. Similarly, the data stage is composed of three packets, the second of which contains varDATA. The third stage employs a handshake packet to report on the status of this transmission. A single 3s1vUTAc process requires transmission of (less than) 32.5 bytes. Assuming a data transfer rate of 480 MBit/s, USB 2.0 is capable of performing over 1,935,832 3s1vUTAc processes per second. A single 3s1vUTAc process will require approximately $0.5us$ ($t_{3s1vUTAc} = 0.5us$).

An overview of the USB-based actuation system for the analog emulator is shown in Fig. 5. An Atmel AT91SAM/AT3X8E microcontroller unit (MCU) is utilized to: 1) interpret the device-side USB drivers, 2) manage communication between USB host and device, 3) perform analog-to-digital (DAC) conversion, as well as 4) appropriately route information to actuation hardware. The 20-bit varNAME parameter is stored in memory and supplied to external hardware via Parallel I/O Controller A; the 12-bit varDATA parameter is stored in memory, converted to an analog signal, and supplied to external hardware via output DAC0. Because the number of analog control inputs (N_{In}) required by the emulator far exceeds the number of DAC's (N_{DAC}) provided by any commercially-available microcontroller, these signals must be generated serially and routed to the appropriate port via 1:N_{In} multiplexer (or equivalent constructed as set of series multiplexing devices). A series of N_{In} sample-and-hold circuits are utilized to "save" each analog control signal for later use.

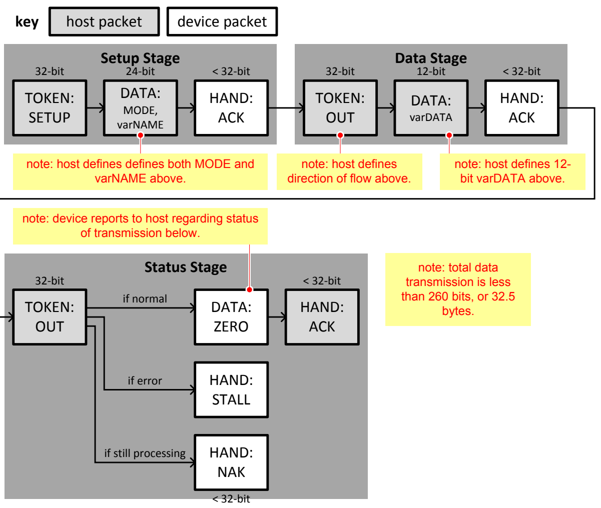

Fig. 4. Three-Stage, Single-Variable, USB-Based Transmission for Actuation Process (3s1vUTAc).

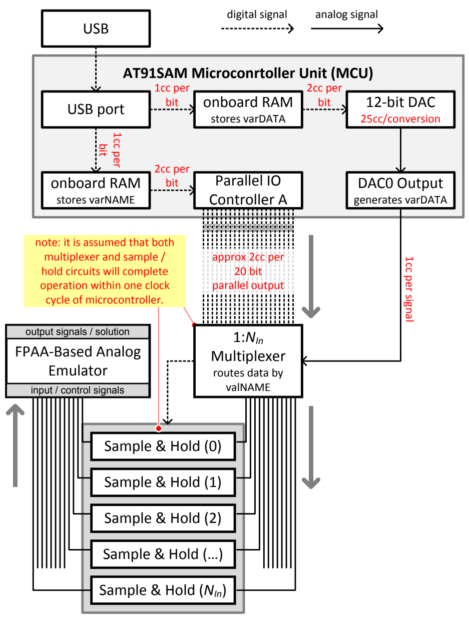
USB
digital signal
analog signal
AT91SAM Microconrtoller Unit (MCU)
USB port
1cc per bit
onboard RAM stores varDATA
2cc per bit
12-bit DAC 25cc/conversion
1cc per bit
onboard RAM stores varNAME
2cc per bit
Parallel IO Controller A
DAC0 Output generates varDATA
note: it is assumed that both multiplexer and sample / hold circuits will complete operation within one clock cycle of microcontroller.
approx 2cc per 20 bit parallel output
1cc per signal
output signals / solution
FPAA-Based Analog Emulator
input / control signals
1:N_{In} Multiplexer routes data by valNAME
Sample & Hold (0)
Sample & Hold (1)
Sample & Hold (2)
Sample & Hold (...)
Sample & Hold (N_{In})

Fig. 5. Overview of USB-Based DAQ Hardware in Actuation Mode.

The Atmel MCU operates with a master clock frequency of 84MHz ($t_{MCUClock} = 12ns$) and provides the user with multiple 12-bit analog-to-digital converters, 12-bit digital-to-analog converters, as well as 32-bit parallel I/O controllers [1]. Documentation provided by Atmel describes the processing time required for certain relevant operations. This MCU requires 1 clock cycle (cc's) to transfer a 32-bit signal from any peripheral to memory and 2 clock cycles to retrieve this information [2009a, 2012b]. Digital-to-analog conversion requires 25 clock cycles for completion [2012b]. Analog-to-digital conversion requires $1us$ for completion [2012b]. The parallel I/O controller requires approximately 2 clock cycles for completion, because it employs a separate $45MHz$ clock (approximately half speed of master). Obviously, these represent best-case-scenario times. In future works, the authors intend to use physical experimentation to more accurately characterize additional overhead.

The time required to properly actuate a single emulator input ($t_{act,i}$) is dependent on that required to perform the 3s1vUTAc process ($t_{3s1vUTAc}$), move varNAME from memory to parallel I/O controller (t_{Move1}), generate the appropriate 20-bit digital control signal (t_{PIO}), move varDATA from memory to the DAC (t_{Move2}), and generate the appropriate analog control signal (t_{DAC}). It is assumed that both multiplexer as well as sample-and-hold circuits are able to complete their tasks is less than one MCU clock cycle. Equations (2) and (3) illustrate that the $t_{Act,i} \approx 896ns$, allowing approximately 1,100,000 actuations per second [2012b].

$$t_{act,i} = \underbrace{t_{3s1vUTAc}}_{500ns} + \underbrace{t_{Move1}}_{\substack{3\ clock\\ cycles}} + \underbrace{t_{Move2}}_{\substack{3\ clock\\ cycles}} + \underbrace{t_{PIO}}_{\substack{2\ clock\\ cycles}} + \underbrace{t_{DAC}}_{\substack{25\ clock\\ cycles}} \quad (2)$$

$$t_{act,i} \approx 500ns + 3(t_{MCUClock}) + 3(t_{MCUClock}) + 2(t_{MCUClock}) + 25(t_{MCUClock}) = 896ns \quad (3)$$

7.2 Data Acquisition

Data acquisition employs a process and hardware very similar to that discussed in Section 7.1 above. The three-stage process shown in Fig. 6 is employed to request data acquisition and transmit relevant information via USB. There are several key differences between the 3s1vUTDa (three-stage, single-variable, USB-based transmission for data acquisition) and 3s1vUTAc processes. Unlike actuation, the data packet in the setup stage defines MODE = 2. This indicates that data acquisition has begun. Within this same packet, the 20-bit varNAME defines the name of the output variable to be acquired as well as, in turn, the origin of this output signal within the emulator itself. Unlike actuation, the token packets within the data as well as status stages indicate that information will flow *IN* from device to host (not *OUT* from host to device). The 12-bit varDATA defines the value of this output variable [2001, 2009b].

Since there is a significantly greater chance of communication failure on the device side, the status stage of 3s1vUTDa is still initiated by the host. This allows this device to indicate if one or more components within the DAQ system have failed or require additional time to complete their tasks. Like the 3s1vUTAc, the 3s1vUTDa process requires transmission of (less than) 32.5 bytes. Making similar assumption as above, a single 3s1vUTDa process will require approximately $0.5us$ ($t_{3s1vUTDa} = 0.5us$).

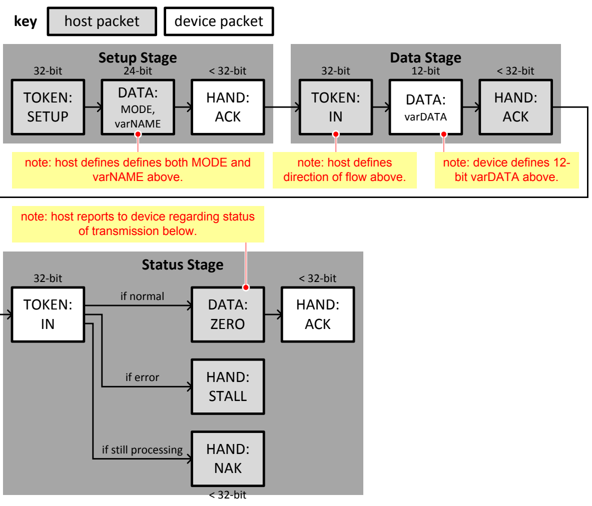

Fig. 6. Three-Stage, Single-Variable, USB-Based Transmission for Data Acquisition Process (3s1vUTDa).

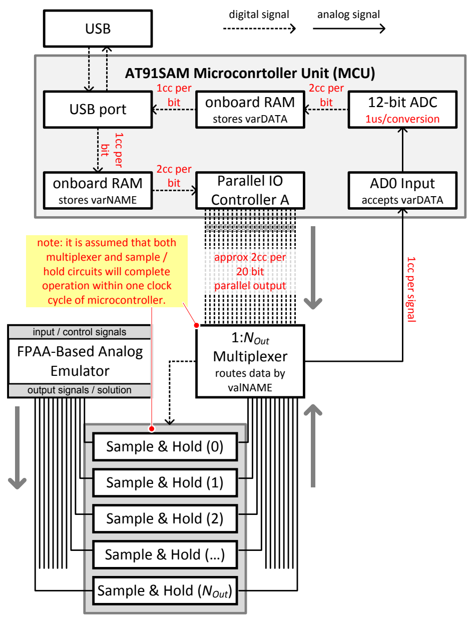
USB
digital signal
analog signal
AT91SAM Microconrtoller Unit (MCU)
USB port
1cc per bit
onboard RAM stores varDATA
2cc per bit
12-bit ADC 1us/conversion
1cc per bit
onboard RAM stores varNAME
2cc per bit
Parallel IO Controller A
AD0 Input accepts varDATA
note: it is assumed that both multiplexer and sample / hold circuits will complete operation within one clock cycle of microcontroller.
approx 2cc per 20 bit parallel output
1cc per signal
input / control signals
FPAA-Based Analog Emulator
output signals / solution
1:N_Out Multiplexer routes data by valNAME
Sample & Hold (0)
Sample & Hold (1)
Sample & Hold (2)
Sample & Hold (...)
Sample & Hold (N_Out)

Fig. 7. Overview of USB-Based DAQ Hardware in Data Acquisition Mode.

An overview of the USB-based data acquisition system for the analog emulator is shown in Fig. 7. It employs a structure that is similar to actuation system shown in Fig. 5. A set of sample-and-hold circuits are used to save the analog emulator's output. A multiplexer is then user to route one of these output signals, as dictated by varNAME, to the device controller for analog-to-digital conversion. The number of output signals to be acquired is referred to as N_{Out}.

The time required to properly acquire a single output from the emulator ($t_{dta,i}$) is dependent on that required to perform the 3s1vUTDa process ($t_{3s1VUTDa}$), move varNAME from memory to the parallel I/O controller (t_{Move1}), generate the appropriate 20-bit digital control signal (t_{PIO}), move varDATA from the DAC to memory (t_{Move3}), and perform analog-to-digital conversion (t_{ADC}). Again, it is assumed that both multiplexer as well as sample-and-hold circuits are able to complete their tasks is less than one MCU clock cycle. Equations (4) and (5) illustrate that the $t_{Act,i} \approx 1596ns$, allowing approximately 600,000 data acquisition operations per second.

$$t_{dta,i} = \underbrace{t_{3s1vUTDa}}_{500ns} + \underbrace{t_{Move1}}_{\substack{3\ clock \\ cycles}} + \underbrace{t_{Move32}}_{\substack{3\ clock \\ cycles}} + \underbrace{t_{PIO}}_{\substack{2\ clock \\ cycles}} + \underbrace{t_{DAC}}_{1us} \quad (4)$$

$$t_{dta,i} \approx 500ns + 3(t_{MCUClock}) + 3(t_{MCUClock}) + 2(t_{MCUClock}) + 1000ns = 1596ns \quad (5)$$

VIII. COMPUTATION TIME

The length of time required to complete a load flow study (t_{study}) is dependent on several factors including: number of load flow analyses required for study (N_i), number of system configuration / emulator input variables (N_{In}), time required to actuate a single emulator input ($t_{acq,i}$), time required for emulation (t_{em}), number of states / emulator output variables (N_{Out}), time required to acquire single emulator output ($t_{dta,i}$), time required or simulation (t_{sim}), and overhead time for pre/post-processing (t_{oh}). Equation (6) defines computation time if the emulator is employed only as a pre-processor, yielding a solution ($\underline{y}_{em,i}$) that is later supplied to simulation engine as an initial condition. Equation (7) defines computation time if simulation is not used, and the emulated solution is used directly to generate study results.

$$t_{study} = t_{oh} + N_i\left(N_{In} t_{acq,i} + t_{emu} + N_{Out} t_{acq} + t_{sim}\right) \quad (6)$$

$$t_{study} = t_{oh} + N_i\left(N_{In} t_{acq,i} + t_{emu} + N_{Out} t_{acq}\right) \quad (7)$$

In either case, the authors hypothesize that the mixed-signal computation engine described in Fig. 3 will generate results faster than simulation alone. Previous works have demonstrated the speed of emulation-based load flow analysis.

IX. REFERENCES

M. R. Dadash-Zadeh, T. S. Sidhu, and A. Klimek, (2009a) "Field-Programmable Analog Array-Based Distance Relay," IEEE Transactions on Power Delivery, vol. 24, p. 10.

V. Salehi, A. Mohamed, and A. Mazloomzadeh, (2012a) "Laboratory-Based Smart Power System, Part I: Design and System Development," IEEE Transactions on Smart Grid, vol. 3, p. 10.

J. R. Arribas, C. Veganzones, F. Blazquez, and C. A. Platero, (2011a) "Computer-Based Simulation and Scaled Laboratory Bench System for the Teaching and Training of Engineers on the Control of Doubly Fed Induction Wind Generators," IEEE Transactions on Power Systems, vol. 26, p. 9.

M. Kayal, R. Cherkaoui, I. Nagel, and L. Fabre, (2007) "Toward a Power System Emulator Using Analog Microelectronic Solid-State Circuits," in IEEE Powertech Conference Lausanne, Switzerland

I. Nagel, R. Cherkaoui, and M. Kayal, (2013) "Analog Microelectronic Emulation for Dynamic Power System Computation," in Department of Electrical and Computer Engineering. vol. Ph.D. Lausanne, Switzerland: Ecole Polytechnique Federale De Lausanne.

A. S. Deese and C. O. Nwankpa, (2011b) "Utilization of FPAA Technology for Emulation of Multiscale Power System Dynamics in Smart Grids," IEEE Transactions on Smart Grid, vol. 2, pp. 606-614.

J. J. Grainger and W. D. Stevenson, (1994) Power System Analysis: Mc-Graw Hill Publishing.

Z. Odibata and S. Momani, (2006a) "Numerical Methods for Nonlinear Partial Differential Equations of Fractional Order," Science Direct Journal of Applied Mathematical Modeling, vol. 32, pp. 28-39.

A. Wood and B. Wollenburg, (1996) Power Generation, Operation, and Control, 2nd ed.: Wiley Publishing.

S. P. Carullo, M. Olaleye, and C. O. Nwankpa, (2004) "VLSI Based Analog Power System Emulator for Fast Contingency Analysis," in Proceedings of 37th Hawaii International Conference on System Sciences (HICSS), Hawaii.

L. Torok and A. Zarandy, (2006b) "Analog-VLSI, Array-Processor-Based, Bayesian, Multi-Scale Optical Flow Estimation," International Journal of Circuit Theory and Applications, vol. 34, pp. 47-75.

R. Fried, R. S. Cherkaoui, C. C. Enz, A. Germond, (1999) and E. A. Vittoz, "Approaches for Analog VLSI Simulation of the Transient Stability of Large Power Networks," IEEE Transactions of Circuits and Systems I, Volume 46, pp. 1249 - 1263.

Atmel Engineering Team, (2012b) "Datasheet for Atmel AT91SAM ARM-Based Flash MCU for SAM3X and SAM 3A Series." Volume 1 United States: Atmel Corporation.

C. Young, M. Devaney, S. Wang, (2001) "Universal Serial Bus Enhances Virtual Instrument-Based Distributed Power Monitoring," IEEE Transactions on Instrumentation and Measurement, Volume 50, Issue 6.

J. Axelson, (2009b) "USB Complete: The Developer's Guide," The Developer's Guide Series, New York City, USA.