A Mixed-Platform Framework for Dynamic Stability Assessment

T. Kyriakidis, G. Lanz, D. Sallin, G. Lilis, L. Fabre, R. Cherkaoui, M. Kayal
Electronics Laboratory, Power Systems Group
Swiss Federal Institute of Technology
CH-1015 Lausanne, Switzerland
theodoros.kyriakidis@epfl.ch

Abstract—**This paper describes a mixed-platform framework dedicated to Dynamic Stability Assessment of power systems. DSA refers to tools capable of characterizing the dynamic stability of the system. Time domain simulation is critical for DSA analysis and is done by algorithms known as TD engines. In this work, operations are shared between a software platform and a hardware one. TD simulation is handled by a dedicated mixed-signal electronics implementation. Data flow control, user interfacing, configuration, result post-processing and other auxiliary operations are realized in software. This architecture combines the flexibility of the software with the high-performance of dedicated hardware. Results of a multi-contingency analysis and a critical clearing time determination analysis for sample test cases are presented. It is demonstrated that an increase in speed of almost three orders of magnitude can be achieved, compared to single-platform solutions.**

Index Terms—**Dynamic Stability Assessment, mixed-platform architecture, mixed-signal emulation, reconfigurable computing.**

I. Introduction

The ongoing major restructuring of power systems has led to them being operated closer to their limits [1], [9]. Therefore stability concerns have been raised, and while security is an inherent function of the system, secure operation can be facilitated by the availability of adequate analytical tools [2].

One such tool is Dynamic Stability Assessment (DSA), which is concerned with the quantitative/qualitative characterization of the ability of the system to retain a state of operating equilibrium after being subjected to severe disturbances. In such transient events examined during DSA studies, the behavior of the system is described by complex, highly non-linear differential-algebraic equations (DAEs), which are handled by specific solvers known as time-domain (TD) engines. The bottleneck of traditional software-only-based solutions lies in the computational complexity of the TD simulation.

This work proposes an alternative mixed-platform architecture. A software platform is responsible for peripheral tasks, and the TD simulation proper is handled by a dedicated hardware platform. Itself, the hardware uses a mixed-signal architecture: algebraic grid equations are solved on an analog resistor lattice, and differential nodal injection equations (generators, loads, etc.) are solved digitally on specially designed reconfigurable hardware. In this way the flexibility of software is combined with the high performance of dedicated hardware.

The rest of the paper is organized as follows. Section II gives the conceptual and theoretical background of the proposed framework. Section III presents the system that has been implemented in the Electronics Laboratory in EPFL. In section IV the application of the latter on a sample test case is showcased, and results are discussed. Finally, V concludes this paper, highlighting the prospect of the proposed architecture.

II. Proposed Architecture

Recent trend in the operation of power systems is that emergency acting is gradually overtaking prevention because of the conservativeness and thereof incurred costs of the latter. Emergency acting suggests for the availability of online and real-time DSA analysis [3]. This requires a massive number of TD simulations to be solved in a small amount of time. It is this computational burden involved in full-system scale DSA that hinders the development of true online real-time implementations. It will be shown in section IV that the proposed architecture lifts the computational bottleneck and enables such approaches.

A. TD simulation fundamentals

The behavior of the power system is governed by a set of parametric differential-algebraic equations. The linear components of the system, such as transmission lines, transformers, and shunt capacitors, form the augmented transmission grid, in red in fig. 1. Non-linear components of the system are nodal injections to the grid, shown in green in fig. 1. Injections can have one, two, or more (rarely)

This work is part of the Innovation Program on Energy of EPFL Middle East. This work is also funded by ABB Switzerland Ltd Corporate Research.

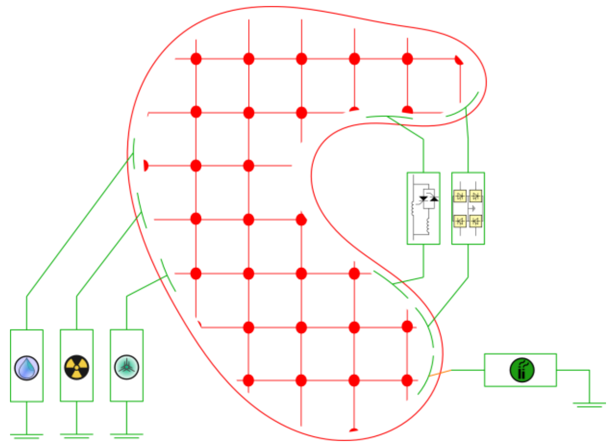


Fig. 1. Abstraction of a power system

endpoints. Examples of injections with one endpoint are generators, dynamic loads etc.; examples of two-endpoint injections are HVDC lines, FACTS etc.

If the study is confined to electromechanical phenomena in the order of Hz, then the fast continuous dynamics of the system, such as electromagnetic phenomena with time constants in the order of milliseconds, can be neglected. Also, given that signals in power systems are periodical and mostly sinusoidal, the phasor representation is used. The system frequency is taken as a reference and signals are expressed as complex quantities in that frame of reference.

With this assumptions, simple algebraic equations can describe the behavior of the grid Y. The latter can be seen as a system where the inputs are complex nodal current injections I and the outputs are nodal voltages magnitudes and angles V.

$$ {}_{(n\times1)}I = {}_{(n\times n)}Y\,{}_{(n\times1)}V \tag{1}$$

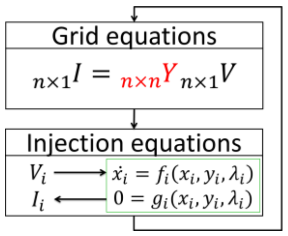
The injections are coupled with the grid in the form of pairs of voltages ("input" to the injection) and current injections ("output" of the dynamic behavior of the injection). The behavior of injection i is described by a set of *state* variables x_i and *instantaneous* variables y_i. The first are governed by differential and the latter by algebraic equations, both of which are parametrized by a parameter set λ_i.

$$S_i = \begin{cases} \dot{x}_i = f_i(x_i, y_i, \lambda_i) \\ 0 = g_i(x_i, y_i, \lambda_i) \end{cases} \tag{2}$$

The complete system of equations (1) and (2) for all i, can be solved in a number of different ways. One of them, following the *partitioned* solution scheme, is presented schematically in fig. 2. At each time step of the integration, the nodal flow algebraic equation of the grid is solved and then the dynamic behavior of the injections is determined.

B. Parallelization

1) Grid equations: Mapping the topology of a power grid on a miniaturized electronic implementation has been proved feasible in [4]-[7] Starting from (1):



Grid equations
$_{n\times1}I = {}_{n\times n}Y\,{}_{n\times1}V$
Injection equations
V_i
I_i
$\dot{x}_i = f_i(x_i, y_i, \lambda_i)$
$0 = g_i(x_i, y_i, \lambda_i)$

Fig. 2. Partitioned solution of power system equations

$$I = Y\,V \Leftrightarrow I_{re} + jI_{im} = (G + jB)(V_{re} + jV_{im}) \tag{3}$$

Neglecting the conductance under the assumption $\|G\| \ll \|B\|$ for the transmission system:

$$\begin{matrix} I_{re} = G\,V_{re} - B\,V_{im} \\ I_{im} = G\,V_{im} - B\,V_{re} \end{matrix} \xrightarrow{G\approx0} \begin{matrix} I_{re} = -B\,V_{im} \\ I_{im} = B\,V_{re} \end{matrix} \tag{4}$$

In (4), B defines two identical separate grids that link voltage to current quantities. These two grids are built using reconfigurable discrete electronic elements.

Inputs to this analog computer are the nodal current injections, outputs are the nodal voltages and the "computing" force is the laws of Ohm and Kirchhoff. This way, immense parallelization is achieved and the solution of the system (1) is available with a complexity of $O(1)$. Comparatively, digital solvers have a complexity of $O(n^3)$ for dense matrices or $O(n^{1.x})$ for favorably structured sparse matrices. Provided that node injections I are handled instantaneously, the speed of this calculation does not depend on the size of the system. Preposterously, it is mentioned here that for the implementation described in section III, the solution of a $I = Y\,V$ system of size up to 96 is attained in a constant time of $t_{sol} = 200\,ns$.

2) Injection equations: In the partitioned scheme of fig. 2, the solution of the equations of each injection is independent from other injections. The latter strongly suggests parallelization of that part of the solution procedure.

The set of injections E is partitioned in K equivalence classes E_k of structurally similar equations. For all i, j members of class E_k, the following holds.

$$\begin{matrix} f_i|_{\lambda_i=\psi_i(\lambda_k)} = f_j|_{\lambda_j=\psi_j(\lambda_k)} \\ g_i|_{\lambda_i=\psi_i(\lambda_k)} = g_j|_{\lambda_j=\psi_j(\lambda_k)} \end{matrix} \quad \forall\, i,j \in E_k \tag{5}$$

Where $\lambda_k, k = 1 \dots K$ is selected so that $\exists\psi_m\colon \lambda_m = \psi_m(\lambda_k) \forall m \in E_k$. The above relation means that differences between DAEs of in-class injections are limited to parameters λ. A simple example for the above is a set of generators modeled using the classical generator model. They are all put under the same equivalence class E_k, and they are all described using a similar set of DAEs (the swing equation), adapted to the different parameters of each individual generator (mechanical starting time and transient impedance).

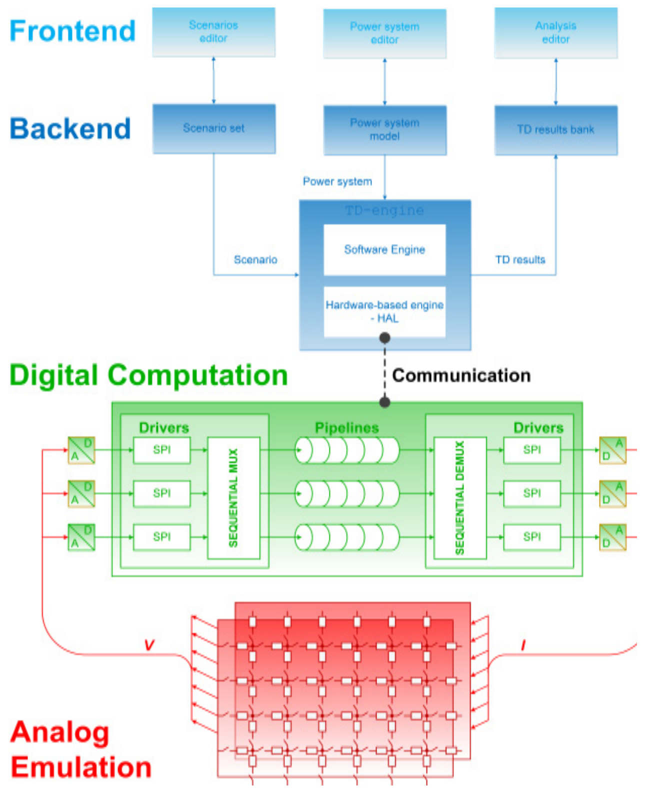


Fig. 3. Architecture of the mixed-platform system

In this work, parallelization according to the above partitioning scheme is suggested: there exists one computational module C_k per equivalence class E_k that is able to process injections of that class.

To further enhance performance, C_k computations are performed in a pipelined fashion. At each pipeline clock cycle the parameters of the system to be solved are iterated across class injections $\lambda_i, i \in E_k$. As a result the system $S_k(\lambda_i)$ is solved, for all the injections of the class. After all pipelined computations are finished, the nodal injections are concurrently updated and grid computations are performed for the next time step, as per section II-B1.

III. Implementation

An implementation of the concept sketched in section II has been realized. Fig. 3 presents the architecture of the system.

A. *Hardware*

The hardware platform, called a Field Programmable Power Network System (FPPNS), is a multi-layered system itself [5]. It consists of *slices* that can be interconnected in a stacked fashion. In this implementation, there are four slices. Each of them has a capacity of 24 power system nodes, giving the system a total capacity of 96 nodes. A photo of the complete hardware can be seen in fig. 5.

Each slice is driven by a dedicated Altera Cyclone III FPGA that implements the digital part of the solver. Pipelines for classical generators, constant impedance, constant current and constant power loads have been created. DAEs in (2) are integrated using a 2nd order linear multistep method with a time-step of as low as $t_{step}^{hw} = 61.035us$. Synchronization between the FPGAs is accomplished via dedicated digital buses.

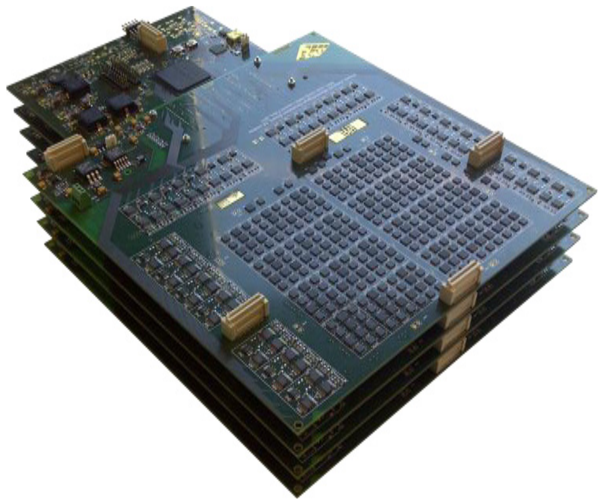


Fig. 5. Photo of the hardware platform

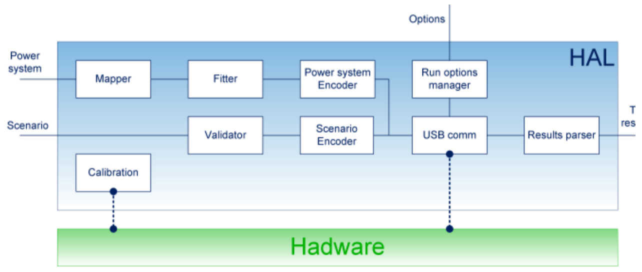


Fig. 4. Software encapsulation of the TD engine based on the dedicated hardware emulator

The analog part is implemented using discrete electronic elements. Reconfigurable resistors and switches allow for the mapping of almost any arbitrary power grid to the topology of the emulator. Two identical resistor lattices are realized, implementing a scaled version of matrix B in (4). Interfacing between the nodes of this analog grid and the FPGA is done using Analog-to-Digital (A/D) and Digital-to-Analog (D/A) converters.

B. *Communication*

The communication layer spans between both platforms. On the hardware side, a Cypress CY7C68013 USB 2.0 controller is interfaced to the FPGA via a two-port RAM. On the software side, a wrapper dynamic-link library (DLL) has been created. The protocol used to communicate commands and data between the two platforms is a flag-polled shared memory scheme. It is shown in the results section IV that communication is the bottleneck of the system.

C. *Software*

A standalone application has been created in C++. This does not aspire to be an EMS-DSA replacement, but is

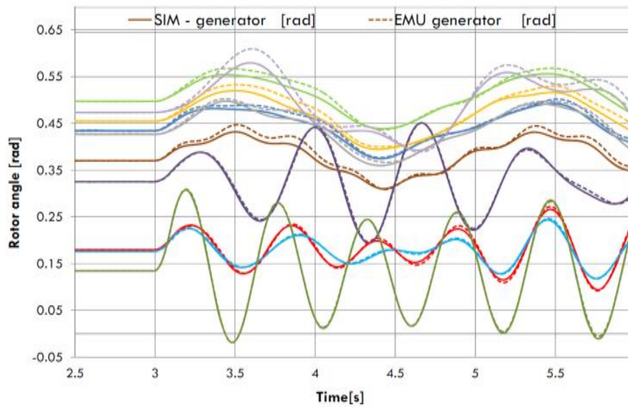


Fig. 5. Generator angles coming from the software (SIM) and the hardware-based (EMU) engines

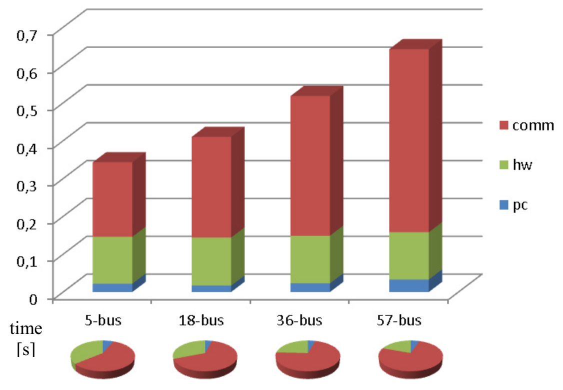


Fig. 6. Timing breakup of a single TD simulation on various system sizes

realized only as a proof of concept. The software can be split in two parts: the frontend (user interface) and the backend (engine). The architecture adheres to the Model-View-Controller (MVC) paradigm.

In the *frontend*, the Graphical User Interface (GUI) is based on the QT framework and the Qwt widget library. On the GUI there are environments dedicated to the creation of the power system, to the definition of the set of scenarios to investigate, and to the analysis of the scenarios. An interface is also provided, where results of various analyses can be real-time graphically superimposed on the schematic of the system under study.

In the *backend*, a `TD-engine` API has been created. `TD-engine` implementations can register to the application, provided they respect the API. Registered engines are fully interchangeable in runtime. Two implementations have been realized: one purely in software and one encapsulating the dedicated hardware of section III-A.

In the software implementation, the set of DAEs is solved in a partitioned fashion, i.e. the solution iterates between differential x and algebraic y variables. The former are handled using a 4-stage explicit Runge-Kutta integration scheme. Resulting linear systems are solved using LU decomposition with partial pivoting. This engine has been created for comparison purposes only.

The `TD-engine` implementation that encapsulates the hardware is actually a Hardware Abstraction Layer (HAL) that respects the `TD-engine` API. A block diagram of it can be seen in fig. 4. The HAL is runtime interconnected to the actual dedicated hardware. A short description of the most important modules is given hereunder.

- The *Calibration* module is responsible for performing a series of automated tests to calibrate the finite-precision reconfigurable elements of the hardware.
- The *Mapper* deals with the graph translation of the real topology of a given arbitrary grid to the constrained topology of the emulator.
- The *Fitter* translates real power system parameters to corresponding scaled parameters of the hardware.
- The *USB communication* module refers to how communication is implemented on the PC side - see also section III-B. Each task assigned to the HAL involves a series of handshaked communication runs with the hardware to complete.

From this point on, the hardware-based `TD-engine` implementation will be referred to as the *emulator*.

D. Limitations

There are a number of limitations in the implementation of section III. The most important ones are listed here.

- Limited number of available nodes
- Neglecting of conductance G of the transmission lines
- Lack of support for a full 3φ model of the grid
- Limited accuracy due to finite precision hardware
- Lack of advanced models of nodal elements
- Support of a narrow spectrum of events

It is the firm intent of the authors to push this concept forward and tackle challenges listed hereinabove.

IV. Results

A demonstration of the use of the proposed architecture is given in this section. Results for sample test systems of 5, 18, 36 and 59 buses are presented.

A. Single TD simulation

The 59-bus system is a modified version of the IEEE 57 bus test system. The response of the system to an indicative 3φ fault has been simulated for 6 s, using a time-step of $t_{step}^{sw} = 10\ ms$ for the software engine and $t_{step}^{hw} = 61.035\ us$ for the emulator. Fig. 5 shows a comparison of machine internal angles coming from the two engines.

Fig. 6 shows the timing break-up of a single emulator run for different system sizes. The pie-charts show how time is divided between the software platform ("pc"), the hardware platform ("hw") and the inter-platform communication ("comm"). As the full set of internal angles of all machines is requested and communicated back to the software, communication requirements increase (linearly) with the number of machines in the system.

It can be easily verified that communication is the performance bottleneck of the proposed architecture. Time spent on software (for preconditioning and other auxiliary tasks) and on hardware (for the computation proper) remains almost the same for all system sizes. The detrimental effect of communication becomes less pronounced when the results requested can be in a compact form. This is normally the case for a multitude of DSA analyses; two examples follow.

B. *Branch contingency analysis*

During an N-1 contingency analysis, the system operator is interested in knowing whether the system retains or not its stability, after an outage/perturbation is applied on each one of its elements. Therefore, in this case the answer from the hardware platform can be communicated with a simple boolean statement for each of the scenarios (stable/unstable), thus greatly reducing communication requirements.

A variation of this procedure was applied to different test systems. A set of branches was selected to be examined and perfect 3φ faults were applied in the middle of each branch for $200\ ms$. The system was simulated for $4\ s$, using a time-step of $t_{step}^{sw} = 10\ ms$ for the software and $t_{step}^{hw} = 244.1\ us$ for the hardware-based engine. Stability was determined based on angle separation of the machines in the system. Table I summarizes the timing results for the software-only (SO) and the mixed-platform (MP) solution.

TABLE I. TIMING RESULTS SUMMARY FOR THE BRANCH CONTINGENCY ANALYSIS

System size	**Branches examined**	**Time SO [s]**	**Time MP [s]**	**Speed up**
5	5	0.9	0.23	3.9 x
18	20	10.2	0.5	20.4 x
36	39	106.7	0.92	116 x
59	55	434.9	1.13	**384.9 x**

C. *Critical Clearing Time determination*

Critical Clearing Time (CCT) determination is concerned with the maximum sustained duration of a fault, for which the system retains its stability. This is usually done using a binary search over a predefined time window. The CCT value is approximated by an upper and a lower bound, the difference of which needs to be under an asked-for precision.

A CCT algorithm similar to the one described here, was implemented on the FPGA. Faults considered were perfect 3φ fugitive faults in the middle of the branches examined in the previous section. A binary search was performed for each fault location, in the search window $t_{src} \in [0.0\ s, 1.5\ s]$, and a precision of $t_{prec} = 10\ ms$ was requested. Timing results for the software-only (SO) and the mixed-platform (MP) architectures are summarized in table II. Simulation options are the same as in the previous section.

It is clear that in both analyses, the performance of the mixed-platform approach is superior by almost three orders of magnitude, compared to the software-only solution. In both cases, results are available online, in exactly the same software environment.

TABLE II. TIMING RESULTS SUMMARY FOR THE CCT ANALYSIS

System size	**Branches examined**	**TD runs**	**Time SO [s]**	**Time MP [s]**	**Speed Up**
5	5	40	7.2	1.13	6.4 x
18	20	160	95.2	3.39	28.1 x
36	39	312	676.8	4.64	145.9 x
59	55	440	4,354.5	11.36	**383.3 x**

Notice that if a larger time-step was selected for the emulator, the time spent on it would be reduced accordingly. In this way much greater speed-ups can be achieved.

V. CONCLUSIONS

The prospect of mixed-platform DSA systems as the one introduced in this paper is very promising. Especially so, as dynamic instability is usually the result of a chain of "black-swan" events, rather than of a single big-scale catastrophic incident [8]. The investigation of multiple contingency scenarios, such as in the form of $n - k$ outages, is enabled with the use of the hybrid architecture described in this work.

Furthermore, the advent of the smart grid reshapes the nature of the transmission and principally of the distribution system. Emerging concepts such as Active Distribution Networks (ADN), microgrids and virtual power plants are self-organizing, self-regulating electrical micro-worlds that stress for distributed intelligence [9]. The pervasiveness of intelligence implied there, strongly suggests the use of dedicated electronics, and computational architectures similar to the one proposed in this work.

REFERENCES

[1] V. Vittal, "Consequence and impact of electric utility industry restructuring on transient stability and small-signal stability analysis," *Proceedings of the IEEE*, vol. 88, no. 2, pp. 196-207, Feb. 1988.

[2] P. Kundur, G. Morison, and L. Wang, "Techniques for on-line transient stability assessment and control," in *Power Engineering Society Winter Meeting, 2000. IEEE*, vol. 1, pp. 46-51.

[3] S. C. Savulescu, *Real-time stability assessment in modern power system control centers*, ser. IEEE Press series on power engineering. Hoboken, N.J.: John Wiley & Sons, Inc., 2009.

[4] L Fabre, I. Nagel, C. Meinen, R. Cherkaoui, and M. Kayal, "A mixed-signal platform dedicated to power system dynamic computation," in *Circuits and Systems (ISCAS), 2011 IEEE International Symposium on,* 2011., pp. 1860-1863.

[5] L. Fabre, I. Nagel, C. Meinen, R. Cherkaoui, and M. Kayal "A field programmable power network system (FPPNS) for high-speed transient stability emulation," in *Power Systems Computation Conference, PSCC'11*, 2011.

[6] L Fabre, G. Lanz, I. Nagel, F. Lo Conte, R. Cherkaoui, and M. Kayal, "An ultra high-speed, mixed-signal emulator for solving power system dynamic equations," in *New Circuits and Systems Conference (NEWCAS), 2011 IEEE 9th International*, 2011, pp. 374-377.

[7] I. Nagel, L. Fabre, R. Cherkaoui, and M. Kayal, "Microelectronic high-speed data processing calculators for power system analysis: Comparison," in *New Circuits and Systems Conference (NEWCAS), 2010 IEEE 8th International*, 2010, pp. 137-140.

[8] G. Andersson, P. Donalek, R. Farmer, N. Hatziargyriou, I. Kamwa, P. Kundur, N. Martins, J. Paserba, P. Pourbeik, J. Sanchez-Gasca, R. Schulz, A. Stankovic, C. Taylor, and V. Vittal, "Causes of the 2003 major grid blackouts in North America and Europe, and recommended means to improve system dynamic performance," *IEEE Transactions on Power Systems*, vol. 20, no. 4, pp. 1922-1928, 2005.

[9] T. Kyriakidis, M. Kayal, "Electronics as a means of facilitating the shift towards the smart grid," *Network Industries Quarterly*, vol 14., no. 2 & 3, pp. 12-16, Jun 2012.