Relaxation-Based Real-time Transient Stability Simulation on Distributed Hardware

Vahid Jalili-Marandi, *Student Member, IEEE*, and Venkata Dinavahi, *Senior Member, IEEE*
Department of Electrical and Computer Engineering
University of Alberta
Edmonton, Alberta, T6G 2V4, Canada
Email: [v_jalili, dinavahi]@ece.ualberta.ca

***Abstract*— Real-time transient stability simulation is of paramount importance for system security assessment and to initiate preventive control actions before catastrophic events such as blackouts happen. Transient stability simulation of realistic-size power systems involves the solution of a large set of non-linear differential-algebraic equations in the time-domain which requires significant computational resources. Exploitation of parallel processing techniques can provide an efficient and cost-effective solution to this problem. This paper proposes a fully parallel method known as instantaneous relaxation (IR) for real-time transient stability simulation. To validate and evaluate the proposed method a test system has been implemented on a distributed PC-Cluster based real-time simulator. A comparison of the captured real-time results with those from the PSS/E software shows high accuracy.**

I. INTRODUCTION

Real-time stability analysis and security assessments is a pressing need in day-to-day operations in the energy control centers. Currently, these analyses are performed off-line because the simulation process for dynamic computation of a realistic size power system takes several hours [1]. Real-time digital simulators are playing an important role in the planning and design of power systems. The application of real-time simulators spans the entire spectrum of traditional power system studies ranging from steady-state to dynamic, and further to high-frequency electromagnetic transient studies. Moreover, when a new device such as a controller or relay needs to be tested and tuned in the hardware-in-the-loop (HIL) scenario for eventual implementation in the field, there is no alternative to a real-time simulator. It should be noted that by real-time we mean hard real-time, i.e. at the end of each time-step, the value of all system variables are available.

Transient stability simulation of realistic-size power systems involves computationally onerous time-domain solution of thousands of nonlinear differential algebraic equations (DAE's). Furthermore, from the point of view of dynamic security assessment which is required for safe system operation and control, several transient stability cases need to be run in a short period of time to initiate preventive control actions. Currently available commercial real-time simulators such as RTDS [2], RT-LAB from OPAL-RT Technologies Inc. [3], and Hypersim [4] address these needs to a large extent by using multiple racks or clusters of multi-processor architectures. These simulators, however, were originally designed and built for electromagnetic transient studies. The sequentiality that exists in the electromagnetic transient computations [5] makes these simulators inefficient (from both cost and performance point of view) for transient stability simulation of large-scale power systems. Moreover, the software API's of the available real-time simulators are closely coupled with their specific hardware technology. For example, RSCAD is the software environment for RTDS, while RT-LAB is OPAL-RT's real-time software manager. These two API's, however, are technology dependent, i.e., RT-LAB cannot run on RTDS hardware, and RSCAD cannot run on OPAL-RT's hardware.

This paper deals with implementation of a new method for real-time transient stability simulation on a distributed PC-cluster hardware. By exploiting parallelism inherent in the transient stability problem as well as the hardware parallelism of the distributed simulator, a parallel solution algorithm is devised to maximize the computational efficiency of the real-time simulator. This reduces the cost of the required hardware for a given system size or increases the size of the simulated system for a fixed cost and hardware configuration. Furthermore, the proposed method can be implemented in such a way to be able to run on any given cluster of parallel computers with only minor changes to be compatible with the hardware requirements.

The proposed approach is a fully parallel *Instantaneous Relaxation* (IR) method for real-time transient stability simulation. The idea of using relaxation-based solution of DAE's is certainly not new and has been explored before. The Waveform Relaxation (WR) method was first introduced in [6] for VLSI circuit simulation. Then in [7] this method was applied to the power systems area and used comprehensively for off-line transient stability simulation [8]. As will be shown later, although the WR method is a parallel method successfully implemented for off-line simulations, there are inefficiencies that surface when it is implemented in real-time. Therefore the IR method which overcomes these limitations is proposed for real-time implementation.

The paper is organized as follows: in Section II a brief overview of the transient stability study in power systems with a discussion about methods for simulation of large-scale systems, and the WR technique is presented. The limitations of the WR method for real-time implementation are also discussed in Section III. The algorithm of the proposed IR method is explained in Section IV, and the approach for partitioning a power system for using the IR method is discussed in Section

V. Practical real-time simulation results and their comparative analysis with off-line simulations using PTI's PSS/E software package are shown in Section VI. Section VII presents the conclusion.

II. TRANSIENT STABILITY SIMULATION METHODS

The differential-algebraic equations (DAE's) which describe the dynamics of a multi-machine power system are given as:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{y}, t) \quad (1)$$

$$\mathbf{0} = \mathbf{g}(\mathbf{x}, \mathbf{y}, t) \quad (2)$$

$$\mathbf{x}(t_0) = \mathbf{x_0} \quad (3)$$

where $\mathbf{x}$ is the vector of state variables, $\mathbf{x_0}$ is the initial values of state variables, and $\mathbf{y}$ is the vector of algebraic variables. The standard method [9] to solve these nonlinear and coupled DAE's involves three steps: the continues-time differential equations are first discretized to be converted to discrete-time algebraic equations, then the non-linear algebraic equations are linearized, and the resulting equations are eventually solved to obtain the system state. As in a realistic power system the size of DAE's is massively large, there have been a lot of efforts to exploit parallel-processing based approaches to reduce the computation time. Diakoptics, parallel-in-time, parallel-in-space and waveform relaxation methods are some of the proposed approaches over the past 50 years. An extensive review of these methods can be found in [10]. The objective in these methods is to basically reduce the computation load on each processor by task-level or program-level parallelization. In task-level approaches the serial algorithm is converted into smaller and independent tasks that can be solved in parallel. It should be noted that the computational efficiency of these methods depends critically on the structure of the system. In the program-level methods, such as waveform relaxation, the parallelism is inherent in the algorithm. In these approaches the system is partitioned into a number of subsystems based on either the system equations or component connectivity. Solving these subsystems is always easier than solving the original system. Therefore, the complexity will be reduced regardless of the system structure.

In the WR method the system is broken into subsystems in a way that the components inside of each subsystem are strongly interdependent while the dependency between components in two different subsystems is weak enough to ignore their interconnection. This characteristic of the WR method makes it suitable for parallel-processing as well as distributed computing. Further details about this method, its convergency conditions, and its application in transient stability study can be found in [11] and [12]. Although the WR method is a parallel method successfully implemented for off-line simulations, there are inefficiencies that surface when it is implemented in real-time.

III. LIMITATIONS OF WAVEFORM RELAXATION FOR REAL-TIME SIMULATION

The outstanding difference between the WR and other classical methods for solving linear and non-linear algebraic equations is that in this method during each iteration each subsystem is analyzed for the entire time interval, $[0, T]$. In other words, elements in this technique are waveforms of the variables rather than their instantaneous values. In each iteration of the WR method each subsystem is solved by using the three basic steps of the standard approach for all $t \in [0, T]$. By using the *windowing* technique in the WR method it is more effective to divide the simulation time into k small intervals or windows, i.e. $[0, T_1]$, $[T_1, T_2]$,..., $[T_k, T]$, and solve equations piece by piece within each interval [11]. Furthermore, windowing reduces the required memory space, because the iterative waveforms need to be stored only for small time intervals instead of the whole simulation time.

The waveform-based property of the WR method is one issue that needs to be changed for real-time simulation. There are two reasons. First, in real-time simulation and specifically in hardware-in-the-loop simulation the instantaneous value of each variable at each time-step is required and not the complete waveforms. Second, if waveforms are going to be used as numerical elements, all waveforms of the variables such as bus voltages or generator angles must be computed for the entire simulation interval, say $20s$, in the first time-step of the simulation, say $1ms$. Clearly, this is not practical for a large-scale system with thousands of variables. To overcome this restriction the windowing technique might be considered, but windowing initiates another obstacle for real-time hardware-in-the-loop simulation. Suppose the simulation interval $[0, T]s$ is divided into k windows of length $m \times h$ milliseconds, h being the time-step. When the simulation starts all waveforms for the interval of the first window must be computed during the first time-step. Then, there are two options. In the first option (Fig. 1) the real-time simulator is idle during the remaining length of the first window, i.e. for $(m-1) \times h$ milliseconds, when it just sends out instant values of variables at each time-step. After this period simulator resumes computation for the interval of the second window, and again becomes idle. This process is repeated until the end of simulation time. In the second option, depicted in Fig. 2, the simulator continues the computation for each window in the subsequent time-steps while it also sends out the instantaneous values of variables at each time-step. Therefore, the computation finishes in k consecutive time-steps, and after that the simulator becomes idle when it sends out instant values at each time-step. It can be concluded that in both options the simulator performs the entire computation in k time-steps and then remains idle for $(m-1) \times k$ time-steps. In other words, the computation load has not been distributed among the time-steps equally. Thus, real-time implementation of the native WR method can be inefficient from resource utilization point of view.

IV. INSTANTANEOUS RELAXATION FOR REAL-TIME TRANSIENT STABILITY SIMULATION

To overcome the limitations of the WR for real-time implementation we propose the *Instantaneous Relaxation* (IR) technique. It is simply the WR method with a window length of one time-step, i.e. $m = 1$. It has been verified in the off-

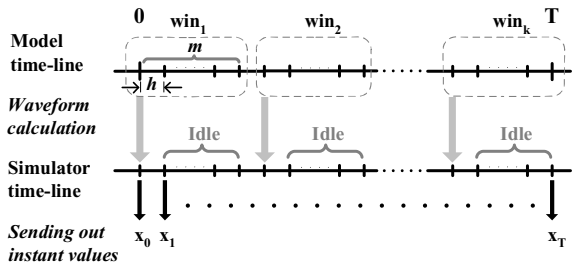

Fig. 1. Real-time implementation of the WR method: Option One.

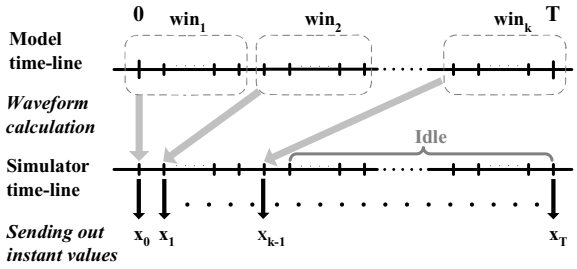

Fig. 2. Real-time implementation of the WR method: Option Two.

line implementation of WR method that the smaller the length of window, the faster the convergence. On the other hand, if the window is made too small the overall communication time among subsystems increases which causes a loss of the advantages of the WR method. However, the communication latency between computation nodes in currently available real-time simulators is in the order of a few microseconds. This latency is small compared to the time-step required for transient stability and can therefore be neglected. Based on the previous experience with the WR method and the arguments made in this paper, it can be concluded that the IR method not only inherits the advantages of the WR method but is also efficient from the real-time simulation point of view.

To apply relaxation methods at the level of differential equations the preliminary step is clustering variables into groups which can be solved independently. This will be specifically discussed for the transient stability application in the next section. After partitioning the system into n subsystems, the set of DAE's equations (i.e. (4) and (5)) are prepared to describe the dynamics of each subsystem:

$$\dot{\mathbf{x}}^i = \mathbf{f}^i(\mathbf{x}^i, \mathbf{V}^i, t) \quad (4)$$

$$\mathbf{0} = \mathbf{g}^i(\mathbf{x}^i, \mathbf{V}^i, t) \quad (5)$$

$$\mathbf{x}^i(t_0) = \mathbf{x_0}^i \quad (6)$$

where $i = 1, 2, \ldots, n$ indicates the subsystem. Discretizing (4) results in a new set of non-linear algebraic equations. In this work we used the trapezoidal rule as the implicit integration method to discretize the differential equations as follows:

$$0 = \frac{h}{2}\left[\mathbf{f}^i(\mathbf{x}^i, \mathbf{V}^i, t) + \mathbf{f}^i(\mathbf{x}^i, \mathbf{V}^i, t-h)\right] - (\mathbf{x}^i(t) - \mathbf{x}^i(t-h)) \quad (7)$$

where h is the integration time-step. (5) and (7) can be linearized by the Newton-Raphson method (for the j^{th} iteration) as:

$$J(\mathbf{z}^i_{j-1}) \cdot \Delta\mathbf{z}^i = -\mathbf{F}^i(\mathbf{z}^i_{j-1}) \quad (8)$$

where J is the Jacobian matrix, $\mathbf{z}^i = [\mathbf{x}^i, \mathbf{V}^i]$, $\Delta\mathbf{z}^i = \mathbf{z}^i_j - \mathbf{z}^i_{j-1}$, and $\mathbf{F}^i$ is the vector of nonlinear function evaluations. (8) is a set of linear algebraic equations that can be solved with Gaussian Elimination and back substitution method. Benchmarking revealed that a majority of execution time in a transient stability simulation is spent for the nonlinear solution. By using the IR method, however, and by distributing the subsystems over several parallel processors, a large-scale system is divided into individual subsystems whose matrix sizes are smaller resulting in faster computations.

Fig. 3 shows the flowcharts of IR method. Practically in the WR method it does not seem efficient to perform several iterations of the Newton-Raphson. Let the exact solution of a waveform be $x(.)$, and the result of the k^{th} iteration of the WR method be $x^k(.)$. Depending on the length of window some iterations will be required for $x^k(.)$ to converge to $x(.)$; however, the starting iterations for $x^k(.)$ are poor approximations of $x(.)$. Thus, it is superfluous to perform Newton-Raphson iterations for computing a close approximation to $x^k(.)$ which itself is a poor approximation of $x(.)$. The convergence rate of the IR method is higher than that of WR, because its window length is minimum. Therefore, performing several iterations of Newton-Raphson, as shown in Fig. 3, increases the accuracy of IR.

Following the convergence of iterative solutions in all subsystems, the state and algebraic variables calculated from the last time-step are updated. The state variables and voltages of generator buses must be exchanged between all interconnected subsystems.

V. SYSTEM PARTITIONING FOR THE IR METHOD

The primary requirement for successfully using the relaxation methods at the level of differential equations, i.e. IR and WR, is to divide the system into subsystems in which tightly coupled variables are grouped together. In [11] it was shown that the WR method will converge for any chosen partitioning scheme; however, the rate of convergence is highly dependent on the method of partitioning [12].

Determination of tightly coupled variables or simply partitioning the system can find a physical meaning from the power system point of view. Following a large disturbance in the system, some generators lose their synchronism with the network. Thus, the system is naturally partitioned into several areas in which generators are in step together while there are oscillations among the different areas. Generators in each of these areas are said to be *coherent*. The coherency characteristic of the power system reflects the level of dependency

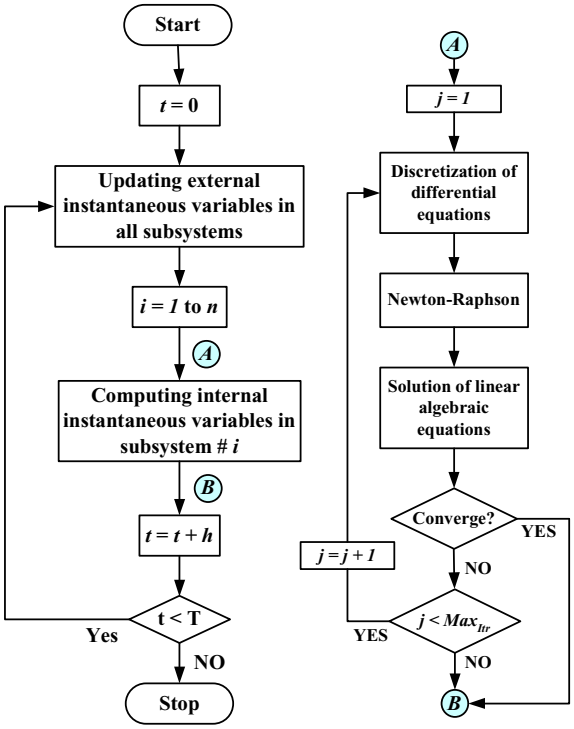

Fig. 3. Flowchart of the proposed IR method for the duration of $[0, T]$ with a time-step of h. j: counter for iterations of the Newton-Raphson; Max_{Itr}: the maximum allowable number of iterations for the Newton-Raphson in each time-step.

between generators. Coherent generators can be grouped in the same subsystem which can be solved independently from other subsystems with the WR or IR methods. The partitioning achieved using the coherency property has two characteristics which make it appropriate for our study. The coherent groups of generators are independent of: (1) the size of disturbance and (2) the level of detail used in the generators. Therefore, the linearized model of the system and the simple classical model of generators can be used to determine coherency. Furthermore, slow coherency based grouping is insensitive to the location of disturbance in the power system [13]. These features of slow coherency lead us to use this partitioning method in this paper.

VI. Experimental Results

In this section we will demonstrate results to verify the efficiency of the IR method for real-time simulation. Several cases with various sizes have been studied, and their real-time results have been validated using the PSS/E software program. The results and discussion for one case is presented here. The case study is the IEEE 39 bus New England test system [14].

A. Real-time simulator structure

A PC-cluster based real-time simulator (Fig. 4) in the RTX-LAB at the University of Alberta is used for the implementation. The *Host* computer running on Windows XP is used as the console for result visualization and online parameter control. The IR method is coded in C and compiled using

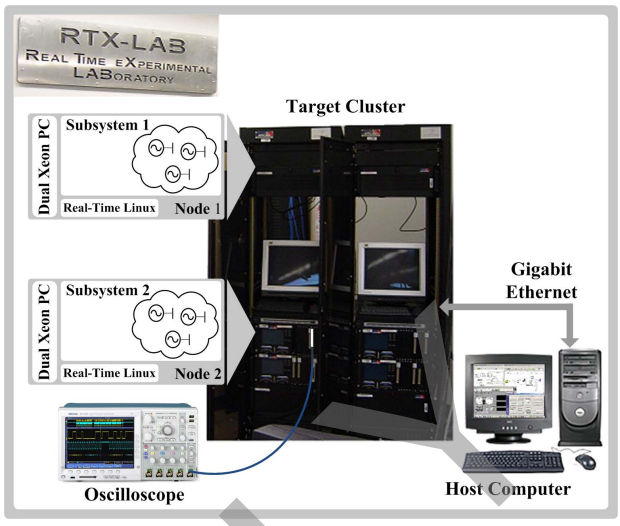

Fig. 4. Configuration of the real-time simulator in the RTX-LAB at the University of Alberta.

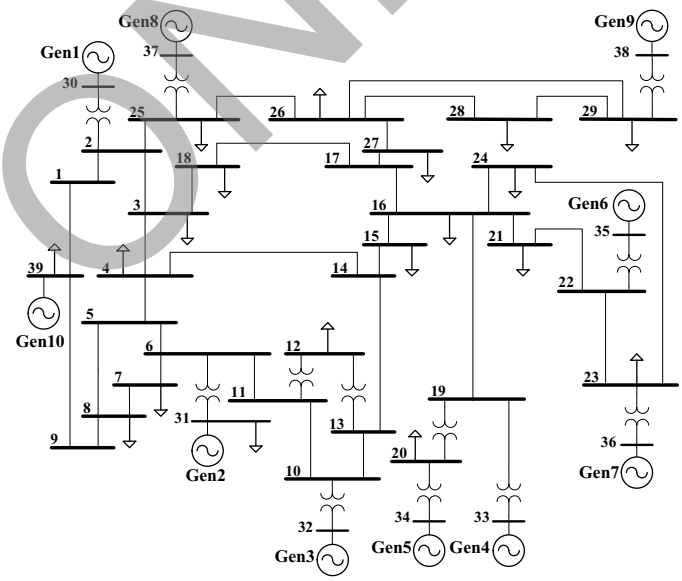

Fig. 5. One-line diagram for the case study.

MATLAB's Real-Time Workshop and OPAL-RT's RT-LAB software [15]. The executable code is loaded onto the *Target Cluster* nodes to run in real-time. Each cluster node consists of a dual $Intel®\ Xeon^{TM}$ shared-memory PC running at $3.0GHz$ on a real-time Linux operating system. The inter-node communication is through InfiniBand with a $10Gb/s$ data transferring rate. The target nodes are capable of eXtreme High Performance (XHP) mode execution, in which one CPU is dedicated entirely to program execution while the other CPU is running operating system tasks and managing the communication schedules with the host computer and other target nodes. The host computer and the target cluster are connected using Gigabit ethernet.

As shown in Fig. 4 a study system can be partitioned and

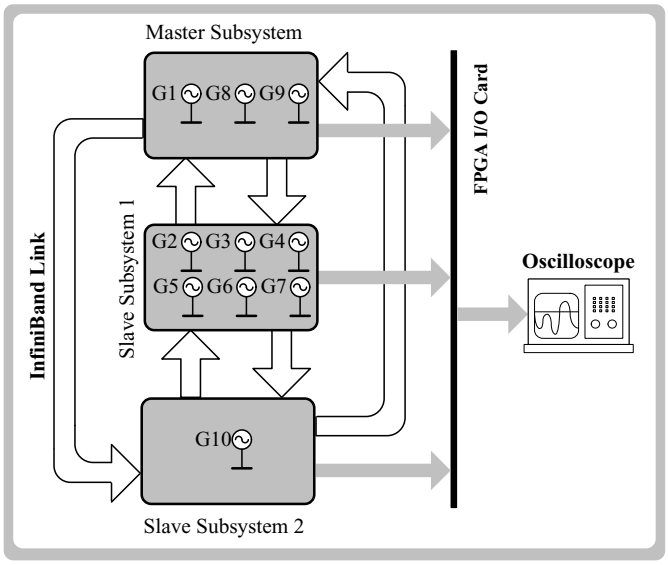

Fig. 6. Distribution of subsystems of the case study among cluster nodes of the real-time simulator.

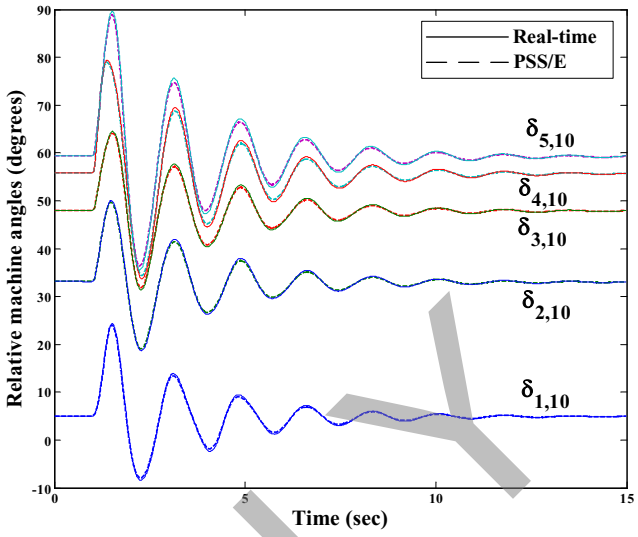

Fig. 7. Comparison of relative machine angles collected from real-time simulator and PSS/E simulation for the case study: $\delta_{i,10} = \delta_i - \delta_{10}$; $i = 1 \ldots 5$.

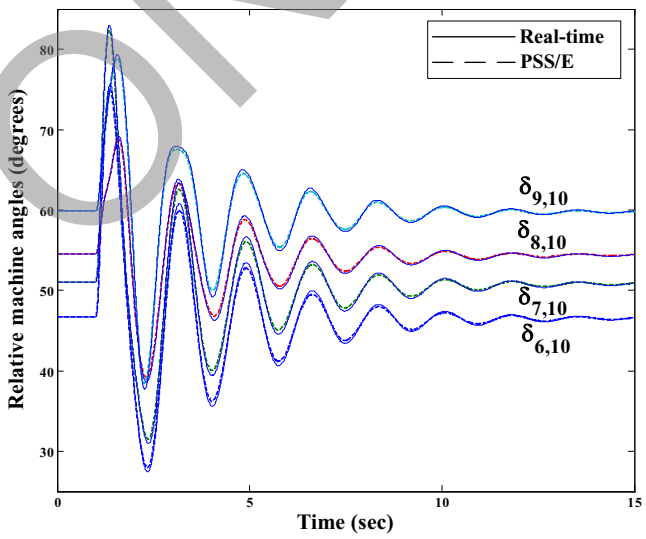

Fig. 8. Comparison of relative machine angles collected from real-time simulator and PSS/E simulation for the case study: $\delta_{i,10} = \delta_i - \delta_{10}$; $i = 6 \ldots 9$.

distributed among the cluster nodes. In the case of using multiple nodes, one node is the Master and other nodes operates as Slaves. The real-time results can be recorded on an oscilloscope connected to the FPGA-based I/O card on the target node or they can be visualized on the host computer screen.

B. Case study

The one-line diagram of IEEE's New England test system is shown in Fig. 5. In this work the detailed model of the synchronous generator including AVR and PSS is used. There are 9 state variables per each generator and 2 algebraic variables per each generator bus [16]. Using the partitioning pattern mentioned in [14], the system has been divided into 3 subsystems: $\{1, 8, 9\}$, $\{2, 3, 4, 5, 6, 7\}$, and $\{10\}$. These 3 subsystems were distributed on three cluster nodes of the real-time simulator: one Master and two Slaves, as shown in Fig. 6. A question which may arise here is about the uneven loading of CPUs. Although it is possible to add $\{10\}$ to subsystem 1 and to use only 2 computation nodes, our intent in this study was to demonstrate the implementation of IR in three parallel cluster nodes. Several fault locations have been tested and the results were compared with those of PSS/E; in all cases results from the IR method match very well. In this section a sample of these results are presented. A three-phase fault happens at Bus 21, at $t = 1s$ and it is cleared after $100ms$. $Gen10$ is the reference generator and the relative machine angles are shown in Fig. 7 and Fig. 8. The maximum discrepancy between real-time simulation and PSS/E was found to be 1.51%, based on (9):

$$\varepsilon_\delta = \frac{max|\delta_{PSS/E} - \delta_{IR}|}{\delta_{PSS/E}} \qquad (9)$$

where $\delta_{PSS/E}$ and δ_{IR} were defined as the relative machine angles from PSS/E and IR method respectively.

Table I shows the timing performance of Master and two Slave nodes in the real-time simulation during one time-step ($1ms$). The sampled idle times were found to be uniform across several time-steps. In the PC-cluster architecture the Master node is responsible for communicating with the host computer and also for organizing the communication among the Slaves. This explains why the Master's communication time in Table I is larger than Slaves' communication time. Moreover, it can be seen that the computation time is not very high, since the computation load is distributed equally among all time-steps. From the idle time duration it is concluded that

TABLE I

PERFORMANCE LOG FOR REAL-TIME SIMULATION OF THE CASE STUDY

	Duration (μs)		
Task	Master	Slave1	Slave2
Computation	212.77	348.13	17.27
Communication	13.44	7.30	4.51
Idle Time	770.93	631.73	974.12
Other	2.86	12.84	4.1
Total Step Size	1000	1000	1000

```
System Initialization ;
Start Timer ;
Do for each time step:
    Repeat until converged:
        Discretizing
        NR & Linear Solution;
        Updating;
    End of Repeat;
End of Do ;
Stop Timer ;
```

```
System Initialization ;
System Partitioning;
Start Timer ;
Do for each time step:
    Solve Subsystem 1;
    Solve Subsystem 2;
    ⋮
    Solve Subsystem n;
    Updating;
End of Do ;
Stop Timer ;
```

Fig. 9. Pseudo code for sequential implementation of standard (left side) and IR (right side) methods for transient stability computations.

larger subsystems can be implemented on each node, and that faster-than-real-time simulation is also possible.

C. Speed-up comparison

In this section we evaluate the speed-up achieved by the IR method in comparison with the standard method. To do so a test program based on the standard approach discussed in Section II was created and its total simulation time was compared with that of the IR method. Since the standard method is sequential, to make a fair comparison, the IR method was also implemented sequentially. Fig. 9 lists steps of these two methods to be run on a $2.5GHz$ quad-core AMD Phenom CPU supported by 4GB of RAM. Except the hardware, the specifications of test case is the same as explained in the previous section. The time-step of both methods is $1ms$. The standard and IR methods total time to simulate a duration of $15s$ is $85s$ and $25s$, respectively. Speed-up is defined as the ratio between the computation time of the standard method to that of the IR method, that for this case is 3.4. The maximum error defined by (9) for both methods is similar. This comparison revealed that on the same simulator hardware, the serial IR method is faster than the standard method for transient stability study.

VII. CONCLUSION

This paper presents a parallel method known as instantaneous relaxation (IR) for the real-time transient stability simulation of power systems. Although it is possible to utilize real-time simulators based on the electromagnetic transient simulation approach to perform transient stability analysis, the size and cost of the simulator is usually prohibitive especially for simulating large-scale systems. The motivation behind this work is to test the real-time feasibility of a fully parallel method that could alleviate these limitations. The waveform relaxation (WR) method was investigated in this paper for implementation in real-time. It was, however, found that WR method has some restrictions for real-time simulation. To demonstrate the performance of the IR method, a case study has been implemented on a PC-Cluster based real-time simulator and the results are validated by the PSS/E software. To verify that the efficiency of the proposed method is not mainly due to the computing capacity of the PC-cluster, a sequential implementation of the IR method was compared with respect to the standard method. This evaluation revealed significant speed-up of the IR method. Slow coherency clustering together with the IR method can lead to an accurate and viable parallel-processor based real-time transient stability simulator package for large-scale systems.

ACKNOWLEDGMENT

Financial support from the Natural Science and Engineering Research Council of Canada (NSERC) is gratefully acknowledged.

REFERENCES

[1] R. Schaniker, P. Miller, W. Dubbelday, P. Hirsch, and G. Zhang, "Real-time dynamic security assessment," *IEEE power and energy magezine,* pp. 51-58, Mar./Apr. 2006.

[2] P. G. McLaren, R. Kuffel, R. Wierckx, J. Giesbrecht, and L. Arendt, "A real time digital simulator for testing relays," *IEEE Trans. Power Del.,* vol. 7, no. 1, pp. 207-213, Jan. 1992.

[3] L.-F. Pak, M.O. Faruque, Xin Nie, and V. Dinavahi, "A versatile cluster-based real-time digital simulator for power engineering research," *IEEE Trans. Power Syst.,* vol. 21, no. 2, pp. 455-465, May 2006.

[4] D. Par, G. Turmel, J.-C. Soumagne, V. A. Do, S. Casoria, M. Bissonnette, B. Marcoux, and D. McNabb, "Validation tests of the hypersim digital real time simulator with a large AC-DC network," *Proc. Int. Conf. Power System Transients,* New Orleans, LA, Sep. 2003, pp. 577-582.

[5] H. W. Dommel, "Digital computer solution of electromagnetic transients in single and multiphase networks," *IEEE Trans. Power App. Syst.,* vol. PAS-88, no. 4, pp. 388399, Apr. 1969.

[6] E. Lelarasmee, A. E. Ruehli, and A. Sangiovanni-Vincentelli, "The waveform relaxation method for time-domain analysis of large scale integrated circuits," *IEEE Trans. Computer-Aided Des. Integr. Circuits Syst.,* vol. 1, no. 3, pp. 131-145, Jul. 1982.

[7] M. Ilic-Spong, M. L. Crow, and M. A. Pai, "Transient stability simulation by waveform relaxation methods," *IEEE Trans. Power Syst.,* vol. PWRS-2, no. 4, pp. 943-949, Nov. 1987.

[8] M. L. Crow, "Waveform relaxation methods for the simulation of systems of differential/algebraic equations with application to electric power systems," Ph.D. dissertation, Univ. Illinois at Urbana-Champaign, Urbana, IL, 1990.

[9] B. Stott, "Power system dynamic response calculations," *Proc. of IEEE,* vol. 67, no. 2, pp. 219-241, Jul. 1979.

[10] J. S. Chai, A. Bose, "Bottlenecks in parallel algorithms for power system stability analysis," *IEEE Trans. Power Syst.,* vol. 8, no. 1, pp. 9-15, August 1993.

[11] J. K. White and A. L. Sangiovani-Vincentelli, *Relaxation techniques for the simulation of VLSI circuits*, Boston, MA: Kluwer, 1987.

[12] M. L. Crow and M. Ilic, "The parallel implementation of the waveform relaxation method for transient stability simulations," *IEEE Trans. Power Syst.,* vol. 5, no. 3, pp. 922-932, Aug. 1990.

[13] H. You, V. Vittal, and X. Wang, "Slow coherency-based islanding," *IEEE Trans. Power Syst.,* vol. 19, no. 1, pp. 483-491, Feb. 2004.

[14] X. Wang, V. Vittal, and G. T. Heydt, "Tracing generator coherency indices using the continuation method: A novel approach," *IEEE Trans. Power Syst.,* vol. 20, no. 3, pp. 1510-1518, Aug. 2005.

[15] Opal-RT Technologies, RT-LAB 8.0.2 [Online]. Available: http://www.opal-rt.com.

[16] P. Kundur, *Power system stability and control*, New York: McGraw-Hill, 1994.