Verifying Noninterference in a Cyber-Physical System – The Advanced Electric Power Grid

Yan Sun[*], Bruce McMillin (Contact)[*†], Xiaoqing (Frank) Liu[*†], David Cape[*]
Department of Computer Science
University of Missouri-Rolla
Rolla, MO 65409
{ysgvd, ff, fliu, dacvdc}@umr.edu

Abstract

The advanced electric power grid is a complex real-time system having both Cyber and Physical components. While each component may function correctly, independently, their composition may yield incorrectness due to interference. One specific type of interference is in the frequency domain, essentially, violations of the Nyquist rate. The challenge is to encode these signal processing problem characteristics into a form that can be model checked. To verify the correctness of the cyber-physical composition using model-checking techniques requires that a model be constructed that can represent frequency interference. In this paper, RT-PROMELA was used to construct the model, which was checked in RT-SPIN. In order to reduce the state explosion problem, the model was decomposed into multiple sub-models, each with a smaller state space that can be checked individually, and then the proofs checked for noninterference. Cooperation among multiple clock variables due to their lack of notion of urgency and their asynchronous interactions, are also addressed.

Type of Submission: Research Paper

Keywords: Interference, Model Checking, Decomposition, Real-time System, Frequency Domain

Relevant Topics: Formal methods: program analysis, model checking, model construction, formal process models, noninterference. Evaluation of software products and components: static and dynamic analysis, validation and verification.

1. Introduction

The term Cyber-Physical System (CPS) describes a system that is intimately coupled with the physical world in space and time and lends itself to hardware/software co-design and co-analysis techniques [1, 2]. This paper discusses the issues in verifying real-time interactions in CPSs using model checking techniques, paying particular attention to the notion of possible interference among the actions of the various components of the system

The CPS under consideration is part of a larger project to design an advanced electric power grid [3]. This is a reliable self-healing grid in which coordinated cyber control of power flow control devices plays a significant role in the system's operation. Flexible AC Transmission System (FACTS) devices are one such device that is used to monitor and adjust the power flows in a power transmission system in real-time. The need for better controls has been shown many times, among which the most serious one is the Northeast blackout 2003 [8]. Mitigating the effects of single contingencies (such as line failures) is necessary before they cause some combination of contingencies which can further lead to a cascading failure scenario in which most or all of a power network goes down. The advanced electric power grid incorporates a number of FACTS devices into a power grid network to act as a distributed, fault-tolerant, and real-time constrained control system to cooperatively adjust power flows.

Two important issues that arise during verification of a CPS include: 1) the interactions among components need to be expressed so that the absence of undesired behavior (interference) can be verified, and 2) the resulting state explosion problem needs to be addressed [6, 11, 14, 15, 16]. The temporal interactions among system components that may cause negative impact are identified and defined during the design process.

* Supported in part by NSF grants CNS-0420869 & CCF-0614633.
† Supported in part by the UMR Intelligent Systems Center.

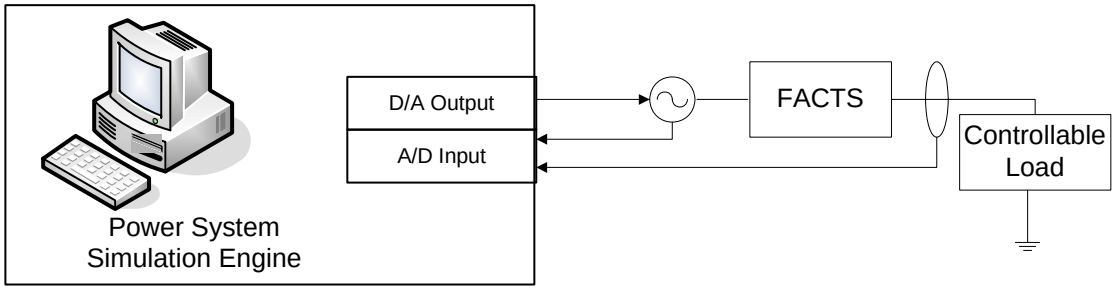

Figure 1. Lab Setup for Hardware-in-the-Loop Test-bed

Within the cyber domain, noninterference is usually defined as the absence of race conditions or absence of violation of component correctness during the composition of system components. To extend this concept to the physical domain requires a different line of thought. The Nyquist rate[1] provides an attractive theory that *bridges* the two domains to relate cyber and physical interactions at the Cyber/Physical boundary in terms of interference between sampling and system frequency response.

Model checking has been used for the verification of hardware and software system designs by specifying a model of the system and its desired properties. In this paper, a decomposition approach was used to break a model into smaller ones. The system is naturally partitioned into a number of high-level models. Only a subset of the properties is checked for each decomposed small model and the resulting proofs are checked for noninterference with overall system correctness. The decomposed models were verified using RT-SPIN [5], a real-time extension of SPIN [13]. RT-SPIN uses RT-PROMELA as the modeling language, which introduces clock variables for real-time control.

This paper is organized as follows. In Section 2, concepts and information related to this study is introduced. This includes the project of building the advanced electric power grid and real-time model checking and related studies. Section 3 contains the description of the problem to be checked. In Section 4, implementation details in the model checking experience such as the important components in the model and the decomposition of the comprehensive model are presented. Issues that were encountered in the model checking are also mentioned. Section 5 contains the verification results. Finally, Section 6 is the conclusion with possible directions for future studies.

2. Model Problem: Advanced Electric Power Grid - HIL Test-bed

The specific system treated in this paper is a laboratory setup of a Hardware-in-the-Loop (HIL) Test-bed of a portion of the advanced electric power grid control system with FACTS devices [1]. Model checking of noninterference among the system components is part of the hardware/software co-analysis and co-design of the HIL test-bed. As shown in Figure 1, the HIL test-bed consists of a real-time simulation of the many lines and buses in an electric power grid, connected to several physical FACTS devices that interact though a few real power lines. The simulation controls the power flow through the power line and interacts with it via A/D Input and D/A Output; the power line flow is set to that of its simulated value. The FACTS devices manipulate the power line at a certain update frequency and the harmonic response is bounded by low-pass filters within the device.

As shown in Figure 2, there are two independent interaction loops in the HIL, one between the FACTS device and the HIL line, the other between the Simulation Engine and the HIL line. Each of the interaction loops involves 1) reading the sensor data and 2) applying the new calculated settings to the HIL line. Both the FACTS device and the Simulation Engine are active components in the system and contain their own computers which take the sensor readings and calculate the new settings. The FACTS device calculates new settings in order to modify the power flow while the Simulation Engine takes the sensor readings and attempts to calculate the response of the entire simulated network and its effects on the current HIL line. The HIL line is an actual power line in the lab setup which only passively responds to the new settings from either the FACTS device or the simulation engine.

Due to the asynchronous nature, possible interference that affects the performance of the HIL Test-bed may

[1] The Nyquist rate from signal processing work is the rate at which sampling of a frequency-capped physical signal such that aliasing does not occur.

exist. Between the cyber and physical (power system) components, this interference is in the frequency domain as follows. The new settings need to be calculated by either the FACTS device or the Simulation Engine based on the correct reading of the current status of the HIL line. In other words, considering that the two interaction loops are asynchronous and new settings from either side can be applied to the HIL line in real time, it needs to be ensured that the sensor reading occurs often enough to capture the changes. If the sampling is too slow (infrequent), some frequency modes are lost and the Nyquist rate is violated – this is defined as interference.

As shown in Figure 2, there are four properties that need to be checked for possible interference: 1) the FACTS reading rate vs. the FACTS setting rate, 2) the Simulation reading rate vs. the Simulation setting rate, 3) the FACTS reading rate vs. the Simulation setting rate, and 4) the Simulation reading rate vs. the FACTS setting rate.

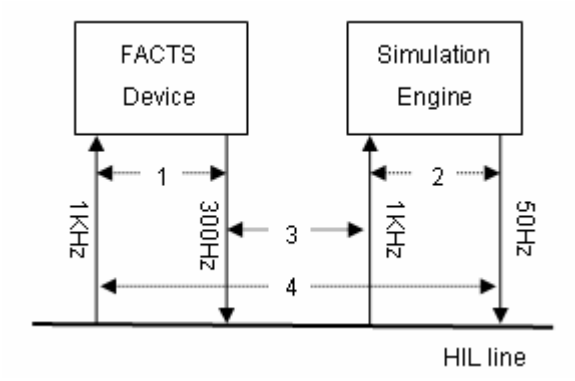

Figure 2. Conceptual model of RT-PROMELA model to be checked

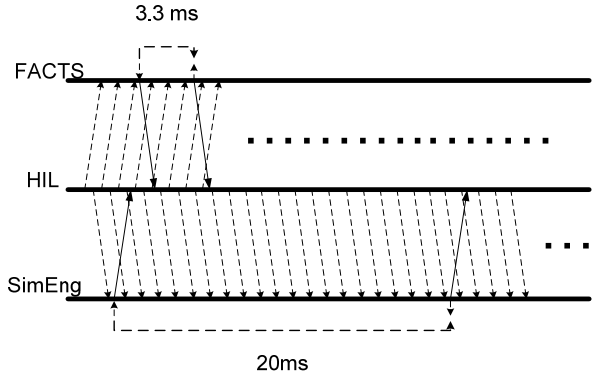

Figure 3. FACTS-HIL and SimEng-HIL Interactions

All four properties are used to ensure that the sensor reading data from the HIL line used in the calculation of new settings contains all of the frequency information present. The interactions between reading and setting from the FACTS device and the Simulation Engine on the same HIL line are illustrated in Figure 3. New settings from the FACTS device are applied to the HIL line every 3.3ms. On the other side, new settings from the Simulation Engine to set the power line to its simulated value are applied to the HIL line at a much slower rate--every 20ms. In order to avoid interference, all four rates above need to satisfy the Nyquist rate.

3. Model Description

Based on the above understanding of the design as shown in Figure 2 and Figure 3, models were developed using RT-PROMELA with assertions to formally check the four properties by defining processes and clock variables. In order to reflect the asynchronous nature of the problem, each of the events should have its own timer. Every process in the model has a clock variable that works with its timer to control its event.

Initially, one Comprehensive model including all the processes and variables was constructed with the hope of checking all four properties at the same time. Due to the state explosion problem of model checking, the model could not be verified using the available computation resources. Thus, model decomposition was applied and four smaller models were derived from the Comprehensive model. They were verified with success. The following subsections will introduce the implementation details of the Comprehensive model and its succeeding decomposition.

3.1. Comprehensive Model

During the construction of the Comprehensive model, in order to keep track of the various rates, the concepts of rendezvous channels and circular buffers were used to record reading and setting events. Each buffer encodes a different frequency response through updates by each of the different clocks, more entries means a faster clock and so on. Rendezvous channels are used for synchronized message passing, which represents how the new settings are applied to the HIL line. The occurrence of an event can be represented by putting a message on the channel. Buffers are used on the HIL line side to receive and record the messages. Each of the four events (two reading events and two setting events) is assigned one channel and one buffer. This notion abstracts [17] the more complex events inside the model checker to simple events relating to frequency interaction,

As shown in Figure 4, a buffer contains *n* slots and a pointer moves along the slots in the buffer at a certain

speed that is no slower than the fastest event. Each time the pointer moves to the next slot, the newly pointed slot clears its value to zero (0). When an event (reading or setting from either the FACTS device or the simulation engine) is sent and received through the channel, the value in the current slot of the corresponding buffer increments by one (1). Otherwise, the current slot does not change its value.

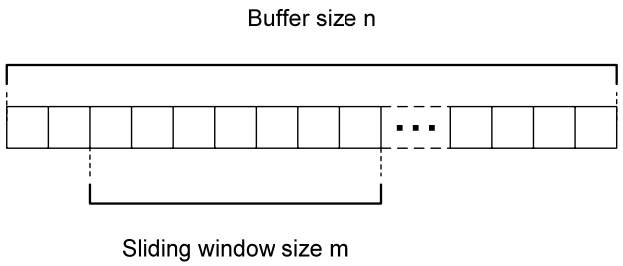

Figure 4. Sliding Window in Model

By keeping the window sizes the same across all four events, we can count the total values in different sliding windows and compare them to make sure that the properties are held under Nyquist rate. In order to check for interference, a sliding window as shown in Figure 4 is used in each of the buffers to count the number of executions of an event over the previous *m* time slots, where m is the size of the sliding window. The values in the sliding windows are counted periodically, for instance, every time the pointer moves to the next slot. In order for the Nyquist rate to hold, the count for the sliding window of a reading event should be at least twice as large as the count for the sliding window of a setting event. The Nyquist rate properties are checked by comparing the counting values for the four events using the following four assertions:

Assertion 1:

*assert(2*cnt_FACTS_affect<=cnt_FACTS_sample);*

Assertion 2:

*assert(2*cnt_FACTS_affect<=cnt_SimEng_sample);*

Assertion 3:

*assert(2*cnt_SimEng_update<=cnt_FACTS_sample);*

Assertion 4:

*assert(2*cnt_SimEng_update<=cnt_SimEng_sample);*

The reading frequencies for both reading events are 1kHz. The FACTS device applies new settings to the HIL line at 1kHz filtered down to 300Hz. The frequency for the Simulation Engine to apply new settings is relatively slow (50Hz) because it involves the changes in the generator as well as the programmable load bank. These numbers are converted into relative values into the RT-PROMELA models and checked with RT-SPIN. The Comprehensive model contains the following components:

Procesess:

- *FACTS_S* (sampling process of the FACT device)
- *FACTS_U* (updating process of the FACTS device)
- *SimEng_S* (sampling process of the simulation engine)
- *SimEng_U* (updating process of the simulation engine)
- *HIL* (the reception process which simulates the HIL line and receives all the events through the channels)

Channels:

- *E_FACTS_Samples_HIL*
- *E_FACTS_Affects_HIL*
- *E_SimEng_Samples_HIL*
- *E_SimEng_Affects_HIL*

Clock Variables:

- *f_s_clock* (the clock variable that controls the FACTS device sampling)
- *f_u_clock* (the clock variable that controls the FACTS device updating)
- *s_s_clock* (the clock variable that controls the simulation engine sampling)
- *s_u_clock* (the clock variable that controls the simulation engine updating)
- *g_clock* (the clock variable that controls the moving of buffer pointers)

Timed automata representing the dynamic behavior and interactions among the five processes were used to help the construction of the model. The events were governed by clock variables and the sensor readings and new settings were represented as synchronization points.

In the model, the sampling and updating events in both the FACTS device and the simulation engine are separated because they are asynchronous and their frequencies are different as shown in Figure 2 and Figure 3. Each channel is designated to one event, either reading or updating, from one of the two active components—FACTS device and Simulation Engine. The clock variables in RT-PROMELA control the

frequencies of the events, one clock variable for each event. The HIL process does not need a clock variable because it is a passive component in the system. All it does is to accept the new settings. The *g_clock* is used to control the speed of the buffer pointer and the sliding window pointer. The *g_clock* variable is put inside the HIL process to handle the task of updating the values in the buffers and managing the pointers movement for the buffers and the sliding windows.

3.2. Model Decomposition

The Comprehensive model contains five clock variables and five processes and we found that the current available computing resources were not adequate in verifying the Comprehensive model because of the well-known state explosion problem in model checking.

The HIL is comprised of a number of high level components. As the models drill down into lower levels, for some of the properties to be checked, not all high level information is required. This decomposition method resembles the traditional abstraction idea [7, 10] and the concept of cluster by Basten et al. [8] to a certain extent. This decomposition combines with the other methods such as symbolic model checking [7, 9]. A model is first decomposed to its finest grain—a Decomposed model does not contain extra processes, variables, etc. in order to verify a subset of properties. The verifications of the decomposed models are then combined to imply the Decomposed model through the noninterference of proofs.

In our experiment, the Comprehensive model is decomposed into smaller models as introduced above to alleviate the state explosion problem. There are four properties/assertions to be checked and each assertion involves the counting numbers for two events—one for the sampling of sensor data and the other for the updating of HIL settings. Each of the four models checks one of the scenarios illustrated in Figure 5. Thus, if one assertion is checked at a time, each smaller model will need to include only three processes, the sampling process, the updating process, and a third one representing the HIL line to receive the messages through channels. Three clock variables are needed--one for each of the two active processes, and a global clock (*g_clock*) for the control of the buffer and sliding window pointers.

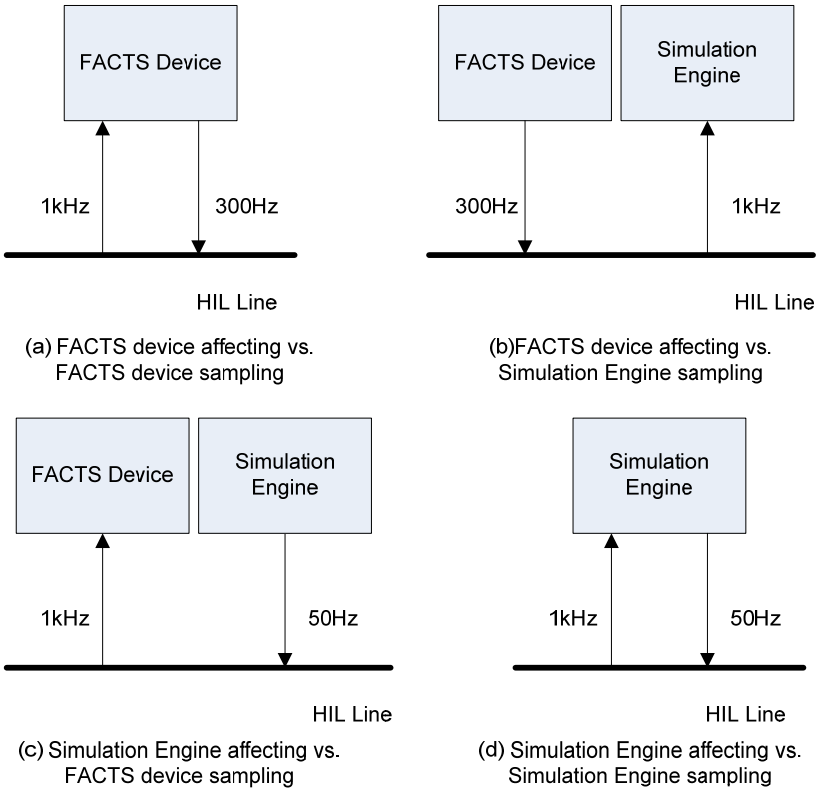

Figure 5. Decomposed Models

In order to further simplify the Decomposed models, the global clock variable for the management of buffer and sliding window pointers can be combined with the faster one of the other two clock variables. In each Decomposed model, there are two clock variables and three processes. For instance, the Decomposed model that involves the sensor reading by the FACTS device vs. applying the new settings from the FACTS device contains the following important components:

Process:

FACTS_S (sampling process of the FACT device)

FACTS_U (updating process of the FACTS device)

HIL (the reception process which simulates the HIL line and receives all the events through the channels)

Channels:

E_FACTS_Samples_HIL

E_FACTS_Affects_HIL

Clock Variables:

f_s_clock (the clock variable that controls the FACTS device sampling and the moving of the buffer pointers)

f_u_clock (the clock variable that controls the FACTS device updating)

Note that the *FACTS_S* process is in charge of the sampling of HIL sensor data and is the faster active process. Thus, its clock variable (*f_s_clock*) also functions as the *g_clock* in the Comprehensive model, which means that *FACTS_S*, in addition to its own task of sending sampling event messages, also needs to update the buffer and manage the pointers.

Among the three processes, the *FACTS_S* process is the most complex one because of its extra responsibility of controlling the buffer and window pointers as mentioned above. The timed automaton for the sampling process is shown in Figure 6. The variable *QUIT* is used to denote acceptable end state because of the way clock variables work in RT-PROMELA. The variable *i* is the pointer variable for the buffer and *j* is the pointer variable for the sliding window.

Figure 7 is the pseudo code for the sampling process on the FACTS device side. It follows the timed automaton in Figure 6. The nested do loop is used to count the numbers of sampling and updating events in the sliding windows. An assertion was added following the completion of counting to check that the sampling frequency is at least twice as high as that of the updating process.

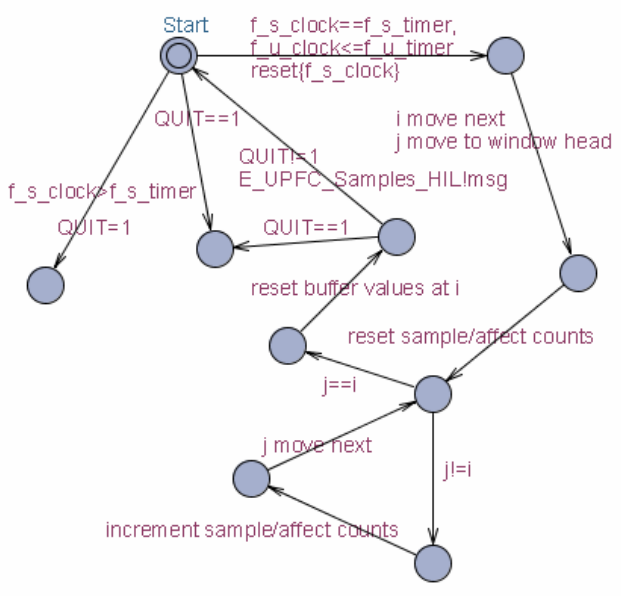

Figure 6. Timed Automaton: FACTS Device Sampling

```
FACTS_S(){
  do
   ::atomic{when{f_s_clock==f_s_timer,
       f_u_clock<=f_u_timer} reset{f_s_clock}
     i=(i+1)%B_SIZE;
     j=(i+(B_SIZE-W_SIZE))%B_SIZE;
     FACTS_Samples_Count=0;
     FACTS_Affects_Count=0;
     do
      ::(j!=i)->FACTS_Samples_Count=FACTS_
          Samples_Count+FACTS_Samples_Buffer[j];
        FACTS_Affects_Count=FACTS_Affects_
          Count+FACTS_Affects_Buffer[j];
        j=(j+1)%B_SIZE
      ::(j==i)->break
     od;
     assert((FACTS_Samples_Count+1)>=2*FACTS_
       Affects_Count);
     FACTS_Samples_Buffer[i]=0;
```

```
        FACTS_Affects_Buffer[i]=0;
        if
          ::E_FACTS_Samples_HIL!msg
          ::(QUIT==1)->break
        fi}
     ::atomic{when{f_s_clock>f_s_timer}
           QUIT=1; break}
     ::(QUIT==1)->break
   od
}
```

Figure 7. Pseudo Code: FACTS Device Sampling

Each of the models directly verifies one of the assertions (1-4) as indicated by (V) in Table 1. However, since the models are verified independently, there is the possibility that the individual verifications do not imply correctness of the Comprehensive model. These possibilities are indicated by (X).

To show that the assumptions made by the verification of the Decomposed Models do not violate the correctness of the Comprehensive Model, we turn, again, to showing noninterference now, solely, in the cyber domain, manually, using the techniques of Owicki and Gries [12].

Lemma 1: The verification of *Model a* with *Assertion 1* does not interfere with the correctness of *Assertion 2*.

Proof:

Using the definition of noninterference where a is an action and assertion is an assertion of interest, we must show:

{pre(a) AND assertion} action a {assertion}

{pre(*Model a*) AND *Assertion 2*} *Model a* {*Assertion 2*}

Since *Assertion 1* is invariant (by the mechanical verification) over *Model a*, we have:

{(*Assertion 1*: 2*cnt_FACTS_affect <= cnt_FACTS_sample)

AND *Assertion 2*: assert(2*cnt_FACTS_affect <= cnt_SimEng_sample)}

Model a {*Assertion2*: assert(2*cnt_FACTS_affect <= cnt_SimEng_sample)}

Since the clocks and sample rate among all four models are fixed, *Model a*'s actions can be considered as assignments

cnt_FACTS_sample = 1000

cnt_FACTS_affect = 300

With the following implication, the assignment axiom holds,

{2*300<=1000) AND (2*300<=cnt_SimEng_sample)}

⇒ (2*300<=cnt_SimEng_sample)}

As long as *Model b* and *Model d* hold cnt_SimEng_sample constant, the implication remains true. □

Theorem 1: The individual verifications of *Models a-d* compose to show the Comprehensive Model.

Proof:

The remainder of the proofs of individual noninterference are symmetric to those of Lemma 1. Since each of *Assertions 1-4* remain invariant over *Model a-d* executions, the conjunction of *Assertions 1-4* remains invariant over the model executions. □

For the Comprehensive Model

Table 1. Assertions affected by Models

	Model a	Model b	Model c	Model d
Assertion 1	V	X	X	
Assertion 2	X	V		X
Assertion 3	X		V	X
Assertion 4		X	X	V

Table 2. Costs of Model Verification

	Buffer Size	Window Size	Depth Reached	# States	Memory Usage
FS_FU	8	6	574011	2.75514e+06	1.91125e+08
FS_SU	10	8	904673	2.4003e+06	1.0977e+09
SS_FU	8	6	574011	2.75514e+06	1.91125e+08
SS_SU	12	10	12817057	3.05495e+07	2.36233e+09

3.3. Other Issues

Some issues were encountered when the model was built. The major issue came from the use of multiple clock variables. The clock variables in RT-PROMELA have no notion of urgency. The laziness prevents the triggering of an event before the clock guard is removed. However, if some other transition is taking place while the clock guard is removed, the chance for the event to be triggered is missed. This laziness problem comes from RT-PROMELA and will affect the model to be checked.

To avoid such a problem, this model also checked for the condition of missing a clock guard. If this happens, the model goes to an acceptable end state, denoted by the variable *QUIT*. This is shown in the timed automaton (Figure 6) and the pseudo code (Figure 7). If the value of the clock variable *f_s_clock* exceeds the clock guard *f_s_timer*, the model sets *QUIT* to one (1) and goes to an end state. This *QUIT* is a global variable and the other processes can check its value. When the other processes find the value of one (1) in *QUIT*, they also go to the acceptable end state. This will early-terminate some of the execution branches during the model checking. These acceptable end states are not a problem with the model but are to avoid the unfavorable effects brought by the laziness of clock variables in RT-PROMELA. Excluding the paths with these acceptable end states will not affect the model checking result. If the other execution paths all satisfy the asserted properties, the model is said to be verified. Otherwise, assertion violation is alerted.

Another issue comes from the way different processes work asynchronously. Because the message recording and the movement of buffer pointers are not synchronized (and this is the desired simulation of the reality), it is possible for two messages to be recorded in the same time slot and leaving one slot empty either before or after the double recording. In this experiment, in order to check the frequency properties, the assertions are modified by introducing an offset of one (1). Thus, the following assertions are used in the models:

*assert(2*cnt_FACTS_affect<=cnt_FACTS_sample+1);*

*assert(2*cnt_FACTS_affect<=cnt_SimEng_sample+1);*

*assert(2*cnt_SimEng_update<=cnt_FACTS_sample+1);*

*assert(2*cnt_SimEng_update<=cnt_SimEng_sample+1);*

4. Model Verification

The Decomposed models were verified using RT-SPIN on the Linux server at Peking University. This server was equipped with 64-bit Power5 microprocessors from IBM and 12 gigabytes of memory. The results showed that based on the current values in the system, no interference was introduced. Table 2 lists the numbers of depth reached and the amount of memory usage (in Bytes) for each of the four Decomposed models. These experiments used the relative timer values based on the numbers in Figure 2. The sampling timers on the FACTS device side and the Simulation Engine side were both set to 6; the timer for updating from the FACTS device was set to 20; and the timer for the updating from the Simulation Engine was set to 120.

The reason to use different buffer and window sizes in different experiments as shown in Table 2 was to find out their effects on the resource usage. These values showed that due to the existence of the state explosion problem, the depth and memory usage grow much faster than the buffer and window size grow. For instance, between the Decomposed models of *SS_SU* and *FS_SU*, the sampling frequencies (*SS* and *FS*) were kept the same and the updating processes were identical. When the buffer size was increased from 10 to 12 and the window size was increased from 8 to 10, the memory usage almost doubled. The number of depth reached and the number of states explored both increased by more than 10 times.

In order to validate the model checking, the timer values were changed to see the effects to the model. For instance, if the reading frequency of the FACTS device is lowered

to less than 600Hz, caused by possible scheduling violation, the Nyquist rate should be violated and possible incorrect readings will cause inaccurate new settings

In response to this scenario, the timer variable for the FACTS reading event was raised from 6 to values above the threshold of 8, such as 10 or 15. As a result, the two models involving the FACTS reading process (FS_SU and FS_SU) reported assertion violation. More similar value adjustments were made in the models and the results were all correct.

5. Conclusion

The advanced electric power grid is a complex real-time system. Thus, the temporal interactions among components need to be carefully examined and verified to avoid interference among them. This paper introduced a successful application of model checking on real-time interference properties using RT-SPIN. The major issue addressed in the experiment is one type of frequency-based interference and the verification of the absence of such interference in the system using model checking. Some other issues related to the asynchronous interactions and modeling language were also discussed, such as the decomposition of the model and the lack of urgency in clock variables in RT-PROMELA.

Model decomposition is one of many ways to tackle the state explosion problem in model checking. In this experiment, the Comprehensive model was decomposed into smaller sub-models. Memory usage was reduced at the cost of more models to be checked. This served our purpose in the experiment by bypassing the state explosion problem. Automated decomposition may be possible. In addition, only one interference scenario was verified in this experiment. With increased semantic understanding of the advanced electric power grid, more interference scenarios can be represented as assertions and integrated with the existing models.

6. References

[1] A. Armbruster, M. Ryan, F. Liu, B. McMillin, and Y. Cheng. "Hardware/Software Co-design for Power System Test Development," *Proc. of 2004 ACM Workshop on Interdisciplinary Software Engineering Research (WISER)*, Newport Beach, CA, Nov. 15, 2004, pp. 83-88.

[2] M. Ryan, S. Markose, Y. Cheng, F. Liu, and B. McMillin, "Structured Object-oriented Co-analysis/Co-design of Hardware/Software for the FACTS Powers System", *Proc. of the 29th Annual IEEE International Conference on Computer Software and Applications (COMPSAC)*, Edinburgh, Scotland, July, 2005.

[3] M. Crow, C. Gill, F. Liu, B. McMillin, D. Niehaus, and D. Tauritz, "Engineering the Advanced Power Grid: Research Challenges and Tasks," *Proc. of Workshop on Research Directions for Security and Networking in Critical Real-Time and Embedded Systems* (affiliated with the 12th IEEE Real-Time and Embedded Technology and Applications Symposium), San Jose, April 4-7, 2006.

[4] D. A. Stuart, M. Brockmeyer, A. K. Mok, and F. Jahanian, "Simulation-verification: biting at the state explosion problem," IEEE Transactions on Software Engineering, Volume 27, Issue 7, July 2001, pp. 599 – 617.

[5] S. Tripakis and C. Courcoubetis. "Extending promela and spin for real time," In *Second International Workshop, Tools and Algorithms for the Construction and Analysis of Systems*, March 1996, pp 329-348.

[6] T. Basten, D. Bosnacki, and M. Geilen, "Cluster-Based Partial-Order Reduction," *Automated Software Engineering*, Vol. 11, No. 4, October 2004, pp. 365-402.

[7] E. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith, "Progress on the State Explosion Problem in Model Checking," *Lecture Notes in Computer Science*, Vol. 2000, 2001, pp. 176-194.

[8] U.S.-Canada Power System Outage Task Force, "Final Report on the August 14th Blackout in the United States and Canada: Causes and Recommendations," https://reports.energy.gov/BlackoutFinal-Web.pdf, April 2004.

[9] K. L. McMillan, *Symbolic Model Checking*, Boston: Kluwer Academic, 1993.

[10] E. Clarke, O. Grumberg, and D. E. Long, "Model Checking and Abstraction," *ACM Transactions on Programming Languages and Systems*, Vol. 16, No. 5, September 1994, pp. 1512-1542.

[11] C. Baier, M., Grosser, and F. Ciesinski, "Partial Order Reduction for Probabilistic Systems," *Proceedings of the First International Conference on the Quantitative Evaluation of Systems. (QEST 2004)*, September 27-30, 2004, pp. 230-239.

[12] S. Owicki and D. Gries, "An axiomatic proof technique for parallel programs" *Acta Informatica*, 6:319-340, 1976.

[13] G. J. Holzmann, The SPIN Model Checker: Primer and Reference Manual, Addison-Wesley, 2003.

[14] E. M. Clarke and O. Grumberg, "Avoiding the State Explosion Problem in Temporal Logic Model Checking Algorithms," *Proceedings of the 6th ACM Symposium on Principles of Distributed Computing*, August 1987, pp. 294-303.

[15] D. A. Stuart, M. Brockmeyer, A. K. Mok, and F. Jahanian, "Simulation-Verification: Biting at the State Explosion Problem," *IEEE Transaction on Software Engineering*, Vol. 27, No. 7, July 2001, pp. 599-617.

[16] R. Alur, R. K. Brayton, T. A. Henzinger, S. Qadeer, and S. K. Rajamani, "Partial-Order Reduction in Symbolic State Space Exploration," *Formal Methods in System Design*, Vol. 18, Issue 2, Special Issue on CAV'97, March 2001, pp. 97-116.

[17] T. Tidwell, C. Gill, and V. Subramonian, "Scheduling Induced Bounds and the Verification of Preemptive Real-Time Systems," Washington University Technical Report, CSE Department, 2007.