Method for partitioning large system models when using latency
insertion method to speed network solution

Yiwei Zhang¹, Jian Shi¹, Jijun Tang¹, Roger Dougal², Blake Langland²
¹Department of Computer Science and Engineering
²Department of Electrical Engineering
University of South Carolina
{zhang33, shi2, jtang, dougal, langland}@engr.sc.edu

Keywords: Latency insertion method, automatic partition,
large network model

Abstract
Real-time simulation of a large system, for example a
large circuit network, is a challenge as it costs a great deal
of computation. To speed up the simulation, the Latency
Insertion Method (LIM) can be applied to divide a large
system model, characterized by a single large conductance
matrix, into a multiplicity of smaller models. Our focus in
this paper is on a method to partition a large network model
to achieve the best solver performance. In this way each
sub-system can be solved much faster than the large model
while high accuracy is still maintained. Also by
incorporating multi-core mechanism, each sub-system can
be solved on one processor or thread in a distributed
computing environment individually so that parallelism can
be achieved.

1. INTRODUCTION
Real-time simulation of a large system has always been
a challenge for several reasons. First of all, the system
model is characterized by a single large conductance matrix.
If the system contains non-linear components, than the
conductance matrix becomes time-dependant, so it needs to
be calculated during each time step. Second, real-time
simulation requires small time steps for accurate results. So
we can see that real-time simulation demands a great deal of
matrix computation which is extremely time-consuming for
a large system.

Recently, an explicit simulation approach called
Latency Insertion Method (LIM) [1] has been proposed to
divide a large system model, characterized by a single large
conductance matrix, into a multiplicity of smaller models. Lalugu [2] describe how LIM could be applied for on-chip
power-grid simulation, but in his work he manually applied
the LIM method to break the system at every node. Meanwhile, Moreira [3], working with electric power
systems, manually inserted LIM objects at hand-picked
locations, and his approach was not amenable to
automation. Our work moves beyond these two by
introducing a method that automatically partition a large
network model based on LIM approach to achieve the best
solver performance. We shall first develop a sub-system
connector that applies the LIM. Then, employ a method that
automatically check the system to see where LIM connector
can be inserted and partition the system into several sub-
systems. Finally, check and merge some sub-systems if
necessary in order to reduce simulation error while still
keeping the simulation efficiency. Multi-core based parallel
computation can also be applied since each sub-system can
be solved independently during each time step. We will
acquire additional speed up by assigning each core to
undertake the calculation of a subsystem simultaneously.

We have applied this method, in the VTB (Virtual Test
Bed) environment, to the simulation of several large-sized
systems with linear and non-linear components. Depending
on the actual structure of the system, the speedup varies, and
we will show the simulation result on a large E-ship system.

2. METHOD FOR PARTITIONING LARGE SYSTEM
The simulation of a network with a large number of
nodes can be a daunting task and remarkable amount of
CPU time may be required to complete the work. To reduce
the simulation time, we shall try to break the system into
some small sub-systems.

2.1. Latency Insertion Method
The LIM is developed to simulate the high-frequency
response of a large network in the time domain. In this
method, a finite formulation is used to update branch
currents and node voltages in a leapfrog manner. The LIM is
readily enabled in networks with latency. A network has
latency if each node in it has a shunt capacitance to ground
and each branch in it has a series inductance. Such networks
can be observed in distributed RLC-based transmission line
circuits.

In Figure 1, a sample circuit is shown for which the
LIM is enabled. The symbols in Figure 1 mean the
following: i and j refer to the nodes; R<sub>ij</sub> and L<sub>ij</sub> are the series
resistance and inductance of the branch between nodes i and
j, respectively; C<sub>ii</sub> and C<sub>jj</sub> refers to the shunt capacitance
from node to ideal ground; V<sub>i</sub>(t) and V<sub>j</sub>(t) refers to the

voltage at node i and j; \( i_{ij}(t) \) refers to the current in the branch between nodes i and j and \( i_i(t) \) refers to the current from node i through capacitor \( C_i \) to ground.Leapfrog scheme is a second-order integration method to solve differential equations. This scheme relies on staggering the voltages and the currents by half a space step and half a time step.

![Figure 1 Typical equivalent circuit to enable LIM](image1)

In the LIM, the transient simulation is accomplished by updating the node voltages and the branch currents at each time step. The conceptual equivalent circuit looks like as shown in Figure 2. The original circuit can be viewed as been separated and branch current can be viewed as a current source. At time \( n \Delta t \), the sub-systems connected at each side can be solved individually. At time \( (n+1/2) \Delta t \), voltage at node i is given by:

\[
v_i^{n+\frac{1}{2}} = v_i^{n-\frac{1}{2}} + \Delta t \frac{C_i}{V_i} i_i^n
\]

And then the branch current at time \( (n+1) \Delta t \) can be derived as following that is based on previous calculation:

\[
i_{ij}^{n+1} = \frac{1 - R_{ij}\Delta t}{R_{ij} + 1} i_{ij}^n + \frac{\Delta t}{1 + R_{ij}\Delta t} \left( v_i^{n+\frac{1}{2}} - v_j^{n+\frac{1}{2}} \right)
\]

The newly calculated branch current will be used for the next time step’s simulation and the above operations will be iterated.

![Figure 2 Conceptual equivalent circuit with LIM](image2)

2.2. Method to partition the system

2.2.1. Making a connector

We use VTB (Virtual Test Bed) system to perform real-simulation of systems. A system is made up of a series of nodes connected by components. VTB will analyze the input system and create the conductance matrix corresponding to the system. Then at each time step, it calculates the inversion of the matrix if the matrix is time-dependent and updates the across and through values on each node.

In first step, we developed a subsystem connector that applies the LIM. The LIM subsystem connector is a two-port element having a set of natural ports on each side. But unlike other two-port elements having natural ports, the two halves are not related to each other by a conductance matrix, so each of the two ports can appear in separate subsystems that are independently solvable. We implement an additional interface for LIM connector so that VTB solver would automatically recognize that whether the nodes connected by LIM belong to the same subsystems or not. If two sub-systems are decoupled, VTB system would instantiate two separate solvers, one for each subsystem.

2.2.2. Method to partition the system

We introduce a method employs a Breadth-First-Search algorithm that is familiar in graph theory. It will automatically identify places that LIM can be applied. A LIM connector can be used if two nodes are connected only by one certain component, for example, a line. Then LIM connectors are used to replace original components and nodes of the original system will be grouped to form different sub-systems. If a LIM connector is within one sub-system, then this LIM connector will be abandoned since it cannot break the system. The algorithm will work as follows:

- Initialize each node as unmarked
- For each node in the system
  - If the node is not marked
    - Mark the node
    - Create a sub-system and put the node into it
    - Get the first node in the sub-system
    - Loop until current node is not NULL
      - For each node that is connected to current node
        - If the nodes are suitable for LIM connection
          - Insert LIM connector
        - Else
          - Add the node to the sub-system and mark it
      - End of if
    - Get next node in the sub-system
- End of Loop
- End of if
- End of for
When the algorithm finishes, all the nodes in the system will be divided into several sub-systems that are connected by LIM connectors. Then in each time step of the simulation, VTB will solve each sub-system separately. After each time step, each LIM connector will update its branch current based on the current calculated result for next time step.

2.2.3. Check and merge the sub-system

If a sub-system contains non-linear component, then the corresponding conductance matrix becomes time-dependent and the inversion of the matrix needs to be computed during each time step. If all the components within a sub-system are linear, then the inversion of conductance matrix only needs to be computed once. Since the inversion of a matrix consumes most part of computation, we shall check each sub-system whether it has a non-linear component or not. If two adjacent sub-systems connected with LIM are both linear, then we shall merge these two sub-systems together. This will not have much effect on the simulation time because only non-linear sub-systems dominate the majority of simulation time. Although the size of the conductance matrix increases when merging two linear systems, the inversion of matrix only needs to be computed. On the other hand, merging the sub-systems will reduce the number of LIM connectors used. This would reduce the error in the simulation.

We put emphasis on the number of LIM components because inserting a LIM object entails adding a shunt capacitance to ground, and simulation results using this method are necessarily not identical to those obtained without the LIM partitions. To some extent, simulation speedup is gained at the expense of accuracy. We would set up a principle whether to merge two adjacent non-linear sub-systems or not. If the speed up -- which can be estimated by the larger number of nodes of each subsystems compared with the total number of nodes of the two sub-systems -- is not larger than some defined threshold, then we merge these two sub-systems.

2.3. Exploiting Multi-core Computing

Since 2005, race between semiconductor manufactures has changed from achieving higher CPU clock frequency to integrating multiple execution cores. This trend provides modern, more powerful multi-core architectures to software developers compares to the previous hardware platforms on which programmers could only use multi-threading to simulate multitasking environment.

We use TPL package (Task Parallel Library) in visual studio 2008 for the parallelizing programming. Comparing to the conventional thread-pool method, TPL has two advantages: Firstly, developers do not need to consider the hardware difference, TPL will automatically take full usage

of the processors installed; Secondly, the code using TPL is much more concise and much less error-prone.

In the current version of our VTB system, multiple solvers are computed sequentially. Since all the sub-systems can be solved independently at each time step, they can be computed simultaneously. We chose this as the start point of applying task-level parallelism and use the Parallel For method in TPL to solve each sub-system concurrently on multiple cores.

3. SIMULATION EXPERIMENT

We use E-ship system for simulation experiment. This system contains 483 nodes and 1248 components. Figure 3 shows the architecture of this system. It has four SSLC parts linked on a ring bus. Most of the components are linear, but it also has non-linear components like generator and diodes in each SSLC part.
original system, only eight of them are replaced by LIM connectors because other sub-systems connected by lines are linear and do not need to be separated.

3.1. Simulation time comparison

The simulation time step was 0.0001 second and the total simulation interval was 0.1 seconds, and the simulation was performed on a two-core computer. First we ran the simulation without using the LIM based partition. Then we ran the simulation with LIM partition enabled.

The simulation time of the first case was 182.235 seconds and 13.172 seconds for the LIM system. This is about 14 times speed up. Such a great improvement in speed is due to the separation of the non-linear sub-systems which have much less nodes than the whole system. Of course, as mentioned above, the speed up is largely dependent upon the structure of the system.

3.2. Simulation accuracy comparison

We have compared the voltage of the voltage sensor in the sub-system near decoupling point

Figure 5: Waveform comparison of the voltage simulated with and without LIM partition

Figure 6: Local waveform comparison of the voltage simulated with and without LIM partition

4. CONCLUSION

This paper presents a method of automatically partitioning the large system based on LIM approach. The experiment result shows that sufficient speed up can be achieved when using this method while the accuracy is still satisfying.

The speed up is also related with the structure of the system. The experiment shows that our method achieves optimized speed up if small size of non-linear sub-system can be separated from the system.

5. REFERENCES


ACKNOWLEDGEMENTS

This work was supported by the US Office of Naval Research under contract N00014-07-1-0686.