A Method for Efficient Simulation of Hybrid Bond Graphs

Indranil Roychoudhury, Matthew Daigle, Gautam Biswas, and Xenofon Koutsoukos
Dept. of EECS and ISIS, Vanderbilt University
Nashville, TN, USA
Email:{indranil.roychoudhury, matthew.daigle, gautam.biswas, xenofon.koutsoukos}@vanderbilt.edu

Pieter J. Mosterman

The MathWorks, Inc.
Natick, MA, USA
Email: pieter.mosterman@mathworks.com

Keywords: Hybrid system simulation, hybrid bond graphs, component-based modeling, block diagrams, incremental causality reassignment

Abstract

The hybrid bond graph (HBG) paradigm is a uniform, multi-domain physics-based modeling language. It incorporates controlled and autonomous mode changes as idealized switching functions that enable the reconfiguration of energy flow paths to model hybrid physical systems. Building accurate and computationally efficient simulation mechanisms from HBG models is a challenging task, especially when there is no *a priori* knowledge of the subset of system modes that will be active during the simulation. In this work, we present an approach that exploits the inherent causal structure in HBG models to derive efficient hybrid simulation models as reconfigurable block diagram structures. We present a MATLAB® Simulink® implementation of our approach and demonstrate its effectiveness using an electrical circuit example.

1 INTRODUCTION

Modeling and simulation are key to the correct design and safe operation of modern engineering systems with a large number of interacting components. Many of these components are *hybrid*, i.e., they combine continuous and discrete behaviors. Hybrid simulation schemes must correctly handle system behavior across discrete mode transitions that involve model reconfiguration and discontinuous updates to the system state variables. Recent research has begun to address the mathematical complexity of hybrid system simulation schemes [1].

The Hybrid bond graph (HBG) [2] language is one such modeling framework suited for modeling the physical processes of hybrid components. HBGs extend bond graphs (BGs) [3], a physics-based modeling language that provides a uniform lumped-parameter, energy-based, topological framework for modeling across multiple physical domains (e.g., electrical, fluid, mechanical, and thermal). HBGs extend BGs by incorporating switching functions that enable the reconfiguration of energy flow paths in the model. This allows for seamless integration of energy-based modeling and model reconfiguration to correctly handle hybrid behaviors of physical processes. In addition, the topological nature of the models facilitates construction of complex hybrid system models by composing component models.

The inherent causal structure in BG models provides the basis for converting BGs to computational models for efficient simulation (e.g., [3, 4]). HBGs extend BGs by introducing controlled junctions that can either permit or inhibit energy transfer. As a result, the causal structure of the HBG model, and hence its underlying computational structure, needs to be recomputed when mode changes occur [2,5]. In general, the subset of system modes that will be active during simulation may be unknown *a priori*, and pre-enumeration of the computational model for all modes is infeasible for large systems. Therefore, we reassign causality and reconfigure the computational model online when mode changes occur.

In our method for efficient simulation of HBG models, we create component-based block diagram (BD) models, where run-time changes in model configuration are handled by reconfiguring the block computations of the model. Causal changes due to junction switches are handled by local propagation, thus avoiding the need for global reassignment of causality at each mode change. We demonstrate the technique by creating reconfigurable BD models in a commercially available simulation environment, MATLAB® Simulink® [6].

The paper is organized as follows. Section 2 first provides a brief overview of the computational issues associated with the HBG modeling paradigm, and then discusses our approach to handling mode changes in HBGs in an efficient manner. Section 3 presents a methodology for implementing the computational structure in MATLAB Simulink. Section 4 places this work in the context of related research in BGs and hybrid system simulation. Section 5 presents conclusions and future work.

2 COMPUTATIONAL MODELS OF HYBRID BOND GRAPHS

Bond graphs (BG) are topological models that capture energy exchange pathways in physical processes [3]. The generic elements in BGs are energy storage (C and I), energy dissipation (R), energy transformation (TF and GY), and input-

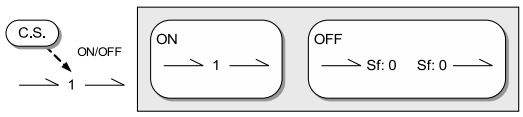

Figure 1. Controlled junction as a Finite State Machine.

output elements (Se and Sf). The connecting edges, called *bonds*, represent energy pathways between the elements. Each bond is associated with two variables: *effort* and *flow*. The product of effort and flow is power, i.e., the rate of energy transfer. Connections in the system are modeled by two idealized elements: 0- (or parallel) and 1- (or series) junctions. For a 0- (1-) junction, the efforts (flows) of all incident bonds are equal, and the sum of flows (efforts) is zero.

Introducing discrete behavior into continuous BGs has been investigated by several researchers [7–9]. Hybrid bond graphs (HBGs) introduce discrete changes in system configuration as idealized switchings of *controlled junctions* [2]. A finite state machine implements the junction *control specification* (CSPEC). Each state of the CSPEC maps to either an *on* or *off* state of the junction, and transitions between states of the CSPEC are functions of system variables and system inputs. When a controlled junction is on, it behaves like a conventional junction. In the off state, all bonds incident on a 1-junction (0-junction) are deactivated by enforcing a 0 flow (effort) at the junction (see Fig. 1). The system mode at any given time is determined by composing the states of the individual switched junctions.

To illustrate the concepts developed in this paper, we will use an electrical circuit example. Figure 2 shows a circuit consisting of a voltage source, $v(t)$, two capacitors, C_1 and C_2, two inductors, L_1 and L_2, two resistors R_1 and R_2, and two switches, SW_a and SW_b. The HBG model for this circuit is given as Fig. 3. The switching junctions in the HBG, 1_a and 1_b, have associated CSPECs, denoted by $C.S._a$ and $C.S._b$, respectively.

2.1 Transforming Hybrid Bond Graphs to Computational Block Diagram Models

The objective of our approach is to derive efficient simulation models from HBG representations. The block diagram (BD) formalism is a graphical, computational scheme for describing simulation models of continuous and hybrid systems, and has been adopted by several mainstream simulation environments, such as Ptolemy [10] and MATLAB Simulink [6]. BD models retain the component-based hierarchical structure of the HBG models they are derived from. Moreover, our ultimate goal is to use the BD simulation models as a testbed for running fault diagnosis experiments [11]. BD models are useful because they preserve the component structure of the model, since the goal in the diagnostic experiments is to isolate the faulty components. Also, BDs allow the introduction of faults by changing parameter values in specific components of the BDs.

Figure 2. Circuit diagram.

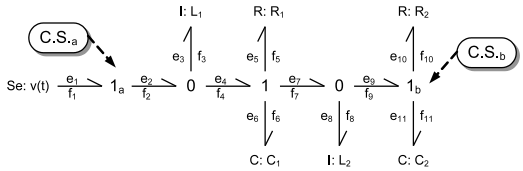

Figure 3. Hybrid bond graph for the example circuit.

In the following, we describe the derivation of BD models from the HBG models of the system. Simulation and analysis of system behavior with BG models is facilitated by the determination of *causality*, i.e., the input-output relationship between the effort and flow variables for each BG element. In this approach, we assume that all components will be in *integral causality*, which means that the computational models for the energy storage elements, (i.e., C and I), are always integral. A standard algorithm for assigning causality to bonds is the Sequential Causal Assignment Procedure (SCAP) [3]. Fig. 4 shows a possible causality assignment for the configuration with both junctions on for the example system.

Fig. 5 shows the BD structure for each BG element [3]. The Sf, Se, C, and I elements have a single unique BD representation because their incident bonds have only one possible causal assignment. The R, TF and GY elements allow two causal representations, each producing a different BD representation. A junction with m incident bonds can have m possible BD configurations. Mapping a junction structure to its BD is facilitated by the notion of a *determining bond*, which concisely captures the causal structure for the junction.

Definition 1 (Determining Bond) *The* determining bond *for a* 0- *(*1-*) junction is the bond that determines the effort (flow)*

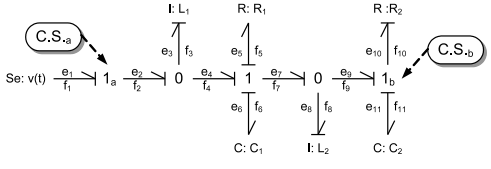

Figure 4. Hybrid bond graph of circuit with both switches on and causality assigned.

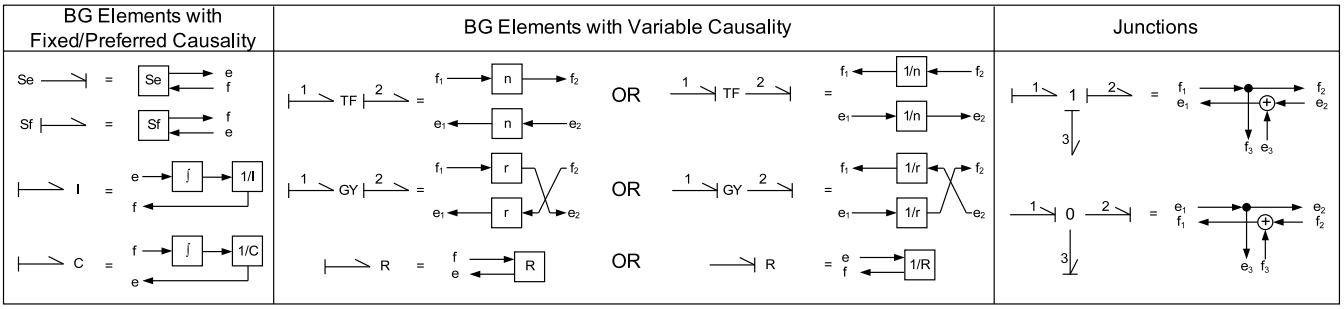

Figure 5. Computational structures for bond graph junctions.

value for that junction.

Fig. 5 shows the BD expansions for 0- and 1- junctions with bond 1 as the determining bond. For a 0-junction (1-junction), all other bonds' effort (flow) values are equal to the determining bond's effort (flow) value, and the flow (effort) value of the bond is the algebraic sum of the flow (effort) values of the other bonds that are connected to this 0- (1-) junction. The determining bond thus plays a crucial role in mapping a HBG to a computational structure.

Converting a causal BG model to a BD is a straightforward procedure. First, each bond is replaced by two signals, i.e., the effort and flow variables for the bond. Next, each junction is replaced by the algebraic constraints they impose (see Fig. 5). The individual blocks for the other elements are now connected using the algebraic constraints imposed by the junctions. The choice of block depends on the assigned causality. Fig. 6 shows the BD representation for the particular mode in Fig. 4.

For HBGs, the BD structure must handle junction switching, and this introduces causal changes, which, in turn, introduce changes in its computational structure. Unlike BGs, the BD model for HBGs must consider multiple forms of BDs for elements whose causality can change.

The changes in the computational structure can be handled in different ways. Given a HBG with n switching junctions, there are 2^n possible junction configurations. All of the corresponding BD configurations can be pre-enumerated off-line, and when junctions switch state, the appropriate configuration can be selected at run-time. However, this requires space exponential in the number of switching junctions. On-line construction of the system BD after junctions switch is space-efficient but wasteful in terms of computation time. Our solution is to construct a structurally adaptable BD model, and to incorporate mechanisms that reconfigure this structure on-line when mode changes occur. Our BD models are structurally static, but they implement local switching within the blocks of associated BG elements. The connections between blocks remain the same, but the interpretation of the signal on the connection (effort or flow) changes depending on the causality associated with the bond in the HBG model.

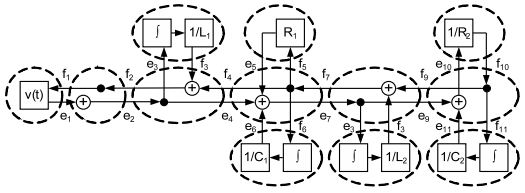

Figure 6. Block diagram representation of circuit with both switches on.

2.2 Efficient On-line Model Reconfiguration

When junction switches occur in a HBG model, the active HBG structure is updated. Changes in the determining bonds of the junctions are evaluated, and these changes are propagated through the model. Since we make the assumption that the system remains in integral causality, we exploit the local propagation of causal changes through the model. This scheme can be combined with a caching mechanism that avoids having to recalculate causal assignment updates for system modes that have occurred previously.

For example, in Fig. 4, if 1_b is switches off, the determining bond of its adjacent 0-junction does not change, and the rest of the BD structure is unchanged. Only the BD representation of 1_b changes. If 1_a switches off, however, the determining bond at the adjacent 0-junction does change and this change propagates step by step to adjacent junctions. In our example, to maintain integral causality, the I element's bond cannot switch causal direction, therefore, bond 4 becomes the determining bond. This change propagates to the adjacent 1-junction, and further up to 1_b, where the R element's bond switches its causal assignment and no further propagation is needed. Fig. 7 shows the causal assignment of the HBG after the mode switch, and Fig. 8 shows the corresponding BD.

At junctions where a unique choice for a new determining bond is not known, arbitrary choices may lead to an inconsistent assignment when the propagation reaches a junction whose determining bond is fixed by an incident source element or an energy-storage element. To prevent such inconsistent assignments, a computationally expensive back-

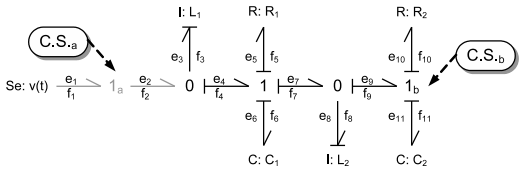

Figure 7. Hybrid bond graph of circuit with switch SW_a open.

tracking process is required. In order to avoid backtracking, we identify active junctions that are in *forced causality* and *fixed causality* and avoid update paths that require determining bond changes for these junctions.

Definition 2 (Forced Causality) *For a given mode of system operation, an active junction is in* forced causality *if its determining bond is uniquely determined.*

Definition 3 (Fixed Causality) *An active junction is in* fixed causality *if, for* all *modes of system operation, its determining bond does not change.*

When a choice of determining bonds exists at a junction, we do not make a choice that would affect the determining bond of an adjacent junction in fixed or forced causality. For example, in Fig. 4, the first two junctions are in forced causality. The other junctions' determining bonds depend on an arbitrary choice of causality assignment to one of the resistors, so they are not in forced causality. In Fig. 7, all active junctions are in forced causality and there is only one consistent assignment of determining bonds for all active junctions.

We formalize this dynamic causality reassignment method as the *Hybrid Sequential Causal Assignment Procedure* (`Hybrid SCAP`, Algorithm 1). Fixed and forced causality information is computed for the initial mode with `Hybrid SCAP` and updated locally when mode changes occur. We assume that the new states of all junctions are available before `Hybrid SCAP` is applied. With the initial queue of switched junctions, `Hybrid SCAP` picks one junction off the queue, and makes all forced changes, and propagates the forced effects using `PropagateEffect` (Algorithm 2) up to junctions that are not forced or fixed. The junction is added to $UnforcedQueue$. When all junctions in the initial queue are exhausted, the algorithm picks elements off the *UnforcedQueue*, assigns a determining bond and propagates its effects till it ends in a junction where another arbitrary choice can be made. If there exists a consistent causality assignment for this mode, the $UnforcedQueue$ eventually becomes empty. Otherwise, the current mode either does not support the integral causality assumption or its HBG model is not well-formed.

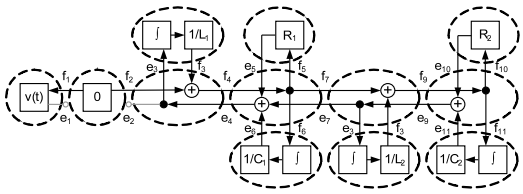

Figure 8. Block diagram representation of circuit with switch SW_a open.

Algorithm 1 Hybrid SCAP

UnassignedJunctionQueue = Set of switched junctions
while *UnassignedJunctionQueue* is not empty **do**
  j = *UnassignedJunctionQueue*.pop()
  if choice of determining bond for j is unique **then**
    Update determining bond of j
    juncList =`PropagateEffect`(j)
    UnforcedQueue.push(*juncList*)
  else
    UnforcedQueue.push(j)
while *UnforcedQueue* is not empty **do**
  j = *UnforcedQueue*.pop()
  if Choice of determining bond for j is unique **then**
    Update determining bond of j
    juncList =`PropagateEffect`(j)
    UnforcedQueue.push(*juncList*)
  else
    if there exists a bond to an unvisited, unforced, unfixed junction to assign as determining bond **then**
      Choose that bond
    else
      Choose bond to a forced junction as determining bond
    Update determining bond of j
    juncList =`PropagateEffect`(j)
    UnforcedQueue.push(*juncList*)

Algorithm 2 PropagateEffect(j)

juncList = [ ]
for all affected adjacent junction *adjJunc* of j **do**
  if choice of determining bond of *adjJunc* is unique **then**
    Update determining bond of *adjJunc*
    juncList+ =`PropagateEffect`(*adjJunc*)
  else
    juncList+ = *adjJunc*
return *juncList*

3 IMPLEMENTATION

In other work, we have developed a physical system modeling environment [12] that supports hierarchical component-based construction of HBG models. Component interfaces are defined by (i) energy ports for energy transfer, and (ii) sig-

nal ports for non-energy related information transfer. Component connections include energy and signal links. We model components as HBG fragments that contain BG elements, modulating functions, and control specifications. The simulation model is created automatically using a model translation procedure, i.e., an *interpreter*, which operates on models created in this environment.

The interpreter creates simulation artifacts from HBG models constructed in the modeling environment. This procedure operates in two steps: (i) the derivation of a BD model from the HBG model, and (ii) generation of simulation artifacts from the BD model. For our implementation, we chose MATLAB Simulink as the simulation environment. The use of the intermediate BD model, however, facilitates easy development of interpreters for several different simulation environments (e.g., Ptolemy) by decoupling the two steps mentioned above. In order to generate these different interpreters for the different simulation environments, only step (ii) will have to be rewritten for the specific target simulation environment. Step (i) will remain the same, thereby giving us considerable savings in the development of new simulation code.

3.1 Translation of Hybrid Bond Graphs to Simulink® Models

Given a HBG model, the first stage of the interpreter navigates the model hierarchy, mapping BG elements to BD elements. The BD modeling language is designed to emulate a generic signal flow diagram. The language consists of the primitives `Blocks` and `Ports`. Hierarchy in the modeling environment is supported by the `Systems` construct, which contains `Blocks`, `Systems` and `Ports`. The hybrid behavior of junctions are also captured through `StateMachines`, which model the CSPECs.

Blocks describe the mapping of their inputs and outputs through a specification. For the purposes of representing a HBG as a BD, specifications describe what kind of BG element a block corresponds to, and gives parameter values of that element. For example, a capacitor's block specification is given as *C(capacitance, initialValue)*. Because the implementation of BG elements can differ from one simulation environment to the other, and because the causality is not captured in this model, the BG elements are not specified in any greater detail.

Since the BD language represents signal flow and BGs represent energy flow, bonds are converted into signals and energy ports into signal ports. Signals are connected through signal ports. At the BD level, the variable passed along a signal connection is not specified, because no causality assignments have been made. The transformation is purely structural. Additional ports are introduced to support the hierarchical, component-based structure of the model.

Because the simulation environment needs to perform causality assignment, a flat HBG data structure is also constructed in this interpretation process. The data structure is essentially a graph describing BG elements and their bond connections. This graph contains the minimal information necessary to assign causality to the BG, i.e., the element types, junction states, and bond connections.

The second stage of the interpretation process involves generating the simulation artifacts from the BD model and the flat HBG data structure. The BD has all the information required to generate the simulation model (the Simulink .mdl file) in its entirety. For every component in the BD model, a corresponding Simulink subsystem with ports and internal blocks is instantiated based on the block specifications. Then these subsystems are connected using the signal paths in the BD model. A library of MATLAB functions (M-code) is generated *a priori* and these instantiations and connections are implemented as calls to these functions.

3.2 Run-Time Implementation

Elements with integral causality (i.e., capacitors and inertias), as well as junctions in fixed causality, are instantiated simply using standard Simulink blocks. In contrast, components with variable causality, i.e., junctions, resistors, gyrators, and transformers, have different implementation equations depending on their causality assignments. Our implementation modifies the BD to handle junction switches and their resulting changes in the computational structure using S-functions [6] to efficiently implement elements with variable causality. S-functions allow for dynamic rerouting of signal flow, therefore, producing a structurally static Simulink model. Individual properties of elements (e.g., number of adjacent bonds and component values) are parameterized in the S-function code so that the implementations are specific only to the element type.

To simulate the system, first, the `Hybrid SCAP` algorithm is run on the MATLAB data structure representing the active HBG structure to obtain an initial causality assignment. The causality information is stored in a global array. Components with variable causality use this information to switch to the correct effort-flow relationships during simulation.

For controlled junctions, the CSPEC is simplified to a two-state machine, having a state each for the on and off state of the junction. We evaluate the guards for the transitions going from the on to off state (the *off-guard*) and the off to on state (the *on-guard*). For example, if the junction is off (on) and the on-guard (off-guard) evaluates to true, the junction switches state. When junctions switch state, the causality of the HBG is reassigned in an incremental manner using the `Hybrid SCAP` algorithm. The S-functions then use these new determining bonds to compute their outputs appropriately.

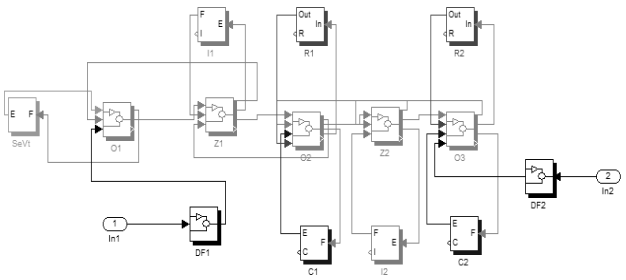

Figure 9. Simulink® model of the circuit example.

3.3 Simulation Results

The Simulink model for the circuit example is shown in Fig. 9 and simulation results are shown in Fig. 10. The simulation starts with junctions 1_a and 1_b on. This corresponds to switches SW_a and SW_b on. 1_a turns off at time step 9 and then turns back on at time step 19. Finally at time step 20, junction 1_b turns off. Thus all three valid configurations[1] of the HBG are visited, and the efforts and flows are plotted, along with the switching junction states. When both controlled junctions are on, the voltage across L_1, denoted by e_2, is equal to the voltage source. The current through the battery $v(t)$, denoted by f_1, increases since it is in parallel to the inductor L_1. The capacitors C_1 and C_2 get charged and the current through them, denoted by f_4 and f_9, decrease. The voltage e_2, across inductor L_1 is equal to that of voltage source and the current through L_2 increases, and hence the voltage e_7 across it decreases. At time step 9, switch SW_a is turned off, disconnecting the voltage source from the circuit, and this results in the discontinuity in the measurements. The results in the different modes can be deduced by simple analysis of the circuit.

[1] A causality assignment cannot be made assuming integral causality when both controlled junctions are in the off state.

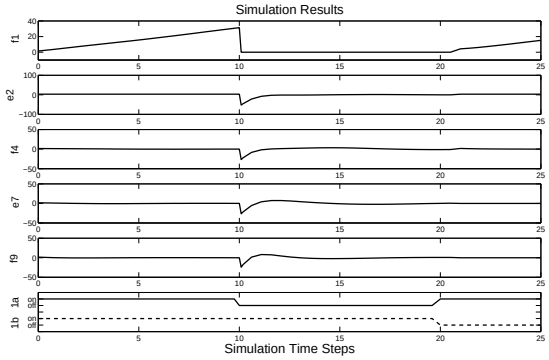

Figure 10. Simulation outputs for a simple control sequence through each discrete state.

4 RELATED WORK

Introduction of discrete behaviors in BGs have been studied by a number of researchers (e.g., [13–16]). Early work used nonlinear resistances [17] and boolean valued modulated transformers [17–20] to include discontinuous behavior. Both of these approaches violated the conditions for idealized lossless switching. [7, 21] introduced an ideal switch, *Sw*, as a new BG element and [22] proposed switching bonds. Our approach adopts controlled junctions [23] to handle idealized mode switches.

Simulation models for dynamic systems are typically expressed as equations (ordinary-differential equations or differential-algebraic equations) [7, 24, 25] and BD models [6, 10, 26]. The discrete switching introduced by HBGs causes the computational models to change when mode transitions occur.

The CAMP-G [19, 20] system compiles equations into code form for execution as MATLAB M-functions or Simulink S-functions. In Dymola [27], the ideal switch can be simply described as:

```
0 = if open then f else e.
```

Bonds with variable causality are implemented as a-causal bonds. The system then generates and simulates an underlying set of differential-algebraic equations [28]. MATLAB Simulink and HYBRSIM [29] can also formulate switching in a similar manner. However, this implementation leads to a much less efficient model since now an algebraic relation solver is required (which could even face convergence problems). For the reasons described earlier, we generate BDs because they retain the topological, component-based, hierarchical structure of the system. Moreover, since many commercial simulation environments, such as Ptolemy [10] and MATLAB Simulink, also use the graphical BD representations, our approach can be easily implemented on any of these target simulation environments.

In addition to the representations used for the simulation models, two other issues play a very important role in the characterization of the hybrid simulation models: (i) the mechanisms employed for determining mode changes during the execution of the simulation, and (ii) the mechanisms employed to recompute the causal structure after a mode change. We discuss these two issues in greater detail below.

Some simulation methods pre-enumerate the simulation model for each mode of operation. This can be done easily if the subset of system modes that are active during simulation is assumed to be known *a priori* [19]. In many situations, for example, when the system uses reconfigurable controllers, or when the mode transitions are autonomous, i.e., they depend on system variables, the active modes cannot be determined before hand. An alternative in this situation is to pre-enumerate the simulation model for all modes (e.g.,

hybrid automata) [25], but pre-enumeration is infeasible for systems with large numbers of possible modes. To overcome this problem, our approach and others (e.g., [10, 24]) build in mechanisms to generate the computational models of an active mode as the execution of the simulation progresses.

In mode-by-mode simulation, the simulation algorithm has to use mechanisms for updating the causal assignments of the BG model in order to determine the model in the new mode. The causal assignment algorithm may recompute the causality assignments in its entirety and regenerate the model (e.g., Dymola [27]) or perform incremental causality assignment and model regeneration. In [24], causality assignment is applied to the entire model at each mode change to generate the new equations. Incremental reassignment is more efficient because in most cases, only a small part of the computational model will change from mode to mode. Therefore, in our approach, we implement incremental causal reassignment, and only change the BD structure in places where the causal assignments change.

We assume integral causality, but there are several approaches that also support derivative causality (i.e., they allow for a change in the model index at run-time), such as, HYBRSIM [29]. HYBRSIM is an experimental application for HBG modeling and simulation. The simulation algorithm includes mechanisms for performing event detection and location based on a bisectional search, and the algorithm can handle run-time causality changes when junctions switch on and off. HYBRSIM runs in interpreted mode and the numerical simulation of continuous-time behavior uses a forward Euler integration algorithm. HYBRSIM can also generate C-code from the designed HBG for compiled simulation. In contrast, our implementation is not interpreted and supports Simulink's variable step solvers.

5 CONCLUSIONS

The work presented in this paper uses physical system modeling semantics as defined by HBGs to impose semantic structure on hybrid computational models in Simulink. Other elegant computational approaches, such as Ptolemy and HyVisual [30] possess these semantics in a mathematical framework, but do not link these semantics to physical system principles. Therefore, we believe that our approach for building computational models from HBGs provides a comprehensive framework for starting from component-oriented physical system models and deriving efficient computational models for hybrid systems. In the future, we will extend our modeling approach and computational model generation schemes to handle situations of derivative causality. We also want to systematically evaluate how our approach performs when applied to real life large physical systems.

ACKNOWLEDGMENTS

This work was supported in part through NSF ITR grant CCR-0225610, NSF grant CNS-0452067, NSF grant CNS-0615214, and NASA USRA grant number 08020-013. Comments by the anonymous reviewers and the help provided by Eric Manders, Chris Beers and Nagabhushan Mahadevan is gratefully acknowledged.

REFERENCES

[1] P. Antsaklis, "A brief introduction to the theory and applications of hybrid systems," *Proc IEEE*, vol. 88, no. 7, pp. 879–887, 2000.

[2] P. J. Mosterman and G. Biswas, "A theory of discontinuities in physical system models," *J Franklin Institute*, vol. 335B, no. 3, pp. 401–439, 1998.

[3] D. C. Karnopp, D. L. Margolis, and R. C. Rosenberg, *Systems Dynamics: Modeling and Simulation of Mechatronic Systems*, 3rd ed. New York: John Wiley & Sons, Inc., 2000.

[4] J. F. Broenink, "20-sim software for hierarchical bond-graph block-diagram models," *Simulation Practice and Theory 7*, vol. 7, no. 5–6, pp. 481–492, 1999.

[5] J. Stromberg, J. Top, and U. Soderman, "Variable causality in bond graphs caused by discrete effects," in *First International Conference on Bond Graph Modeling (ICBGM '93)*, ser. SCS Simulation Series, vol. 25, no. 2, 1993, pp. 115–119.

[6] MATLAB®/Simulink®, "http://www.mathworks.com/products/simulink/."

[7] J. Buisson, H. Cormerais, and P.-Y. Richard, "Analysis of the bond graph model of hybrid physical systems with ideal switches," *Proc Instn Mech Engrs Vol 216 Part I: J Systems and Control Engineering*, pp. 47–63, 2002.

[8] M. Magos, C. Valentin, and B. Maschke, "Physical switching systems: From a network graph to a hybrid port hamiltonian formulation," in *Proc IFAC conf Analysis and Design of Hybrid Systems*, Saint Malo, France, June 2003.

[9] J. van Dijk, "On the role of bond graph causality in modelling mechatronic systems," PhD Dissertation, University of Twente, The Netherlands, 1994.

[10] J. Buck, S. Ha, E. A. Lee, and D. G. Messerschmitt, "Ptolemy: a framework for simulating and prototyping heterogeneous systems," *Readings in hardware/software co-design*, pp. 527–543, 2002.

[11] P. J. Mosterman and G. Biswas, "Diagnosis of continuous valued systems in transient operating regions," *IEEE Trans. Syst., Man Cybern. A*, vol. 29, no. 6, pp. 554–565, 1999.

[12] E.-J. Manders, G. Biswas, N. Mahadevan, and G. Karsai, "Component-oriented modeling of hybrid dynamic systems using the Generic Modeling Environment," in *Proc of the 4th Workshop on Model-Based Development of Computer Based Systems*. Potsdam, Germany: IEEE CS Press, Mar. 2006.

[13] R. Cacho, J. Felez, and C. Vera, "Deriving simulation models from bond graphs with algebraic loops. the extension to multibond graph systems," *J Franklin Institute*, vol. 337, pp. 579–600, 2000.

[14] W. Borutzky, J. Broenink, and K. Wijbrans, "Graphical description of physical system models containing discontinuities," in *Modelling and Simulation, Proc. of the European Simulation Multiconference*, A. Pave, Ed., Lyon, France, June 1993, pp. 208–214.

[15] U. Söderman, J. Top, and J. Stromberg, "The conceptual side of mode switching," in *Proc. System, Man and Cybernetics*, 1993.

[16] F. Lorenz and H. Haffaf, "Combinations of discontinuities in bond graphs," in *Proc. Intl. Conf Bond Graph Modeling Simulation*, Las Vegas, NV, Jan. 1995, pp. 56–64.

[17] D. Karnopp and R. C. Rosenberg, *Analysis and Simulation of Multiport Systems*. New York: John Wiley and Sons, 1975, iSBN 0-471-45940.

[18] J. Garcia, G. Dauphin-Tanguy, and C. Rombaut, "Bond graph modeling of thermal effects in switching devices," in *Proc. Intl. Conf Bond Graph Modeling Simulation*, ser. Simulation, F. E. Cellier and J. J. Granada, Eds., no. 1, Society for Computer Simulation. Las Vegas: Simulation Councils, Inc., Jan. 1995, pp. 145–150, volume 27.

[19] J. J. Granda, G. Dauphin-Tanguy, and C. Rombaut, "Power electronics converter-electrical machine assembly bond graph models simulated with CAMP/G-ACSL," in *IEEE International Conference*, France, October 1993.

[20] J. Granda, "The role of bond graph modeling and simulation in mechatronic systems, and integrated software tool: CAMP-G, MATLAB-simulink," *Mechatronics*, vol. 12, pp. 1271–1295, 2002.

[21] J.-E. Strömberg, J. Top, and U. Söderman, "Variable causality in bond graphs caused by discrete effects," in *Proceedings of the International Conference on Bond Graph Modeling*, San Diego, California, 1993, pp. 115–119.

[22] J. F. Broenink and K. C. Wijbrans, "Describing discontinuities in bond graphs," in *Proceedings of the International Conference on Bond Graph Modeling*, San Diego, California, 1993, pp. 120–125.

[23] P. J. Mosterman and G. Biswas, "Behavior generation using model switching a hybrid bond graph modeling technique," in *Proc. Intl. Conf Bond Graph Modeling Simulation*, ser. Simulation, F. E. Cellier and J. J. Granada, Eds., vol. 27, number 1. Las Vegas: Simulation Councils, Inc., Jan. 1995, pp. 177–182.

[24] K. Edström, "Simulation of mode switching systems using switched bond graphs," Ph.D. dissertation, Linköpings Universitet, Dec. 1996.

[25] K. Edström and J. Strömberg, "Aspects on simulation of switched bond graphs," *Proc. of the 35th Conf. on Decision and Control*, 1996.

[26] C. D. Beers, E.-J. Manders, G. Biswas, and P. J. Mosterman, "Building efficient simulations from hybrid bond graph models," in *IFAC Conference on analysis and design of hybrid systems*, Alghero, Italy, June 2006.

[27] Dymola, "http://www.dynasim.com/dymola.html."

[28] F. Cellier and R. T. McBride, "Object-oriented modeling of complex physical systems using the dymola bond-graph library," in *International Conference on Bond Graph Modeling and Simulation*, Orlando, FL, Jan. 2003.

[29] P. Mosterman, "HYBRSIM - a modeling and simulation environment for hybrid bond graphs," *Journal of Systems and Control Engineering - Part I*, vol. 216, 1, pp. 35–46, 2002.

[30] C. Hylands, E. A. Lee, J. Liu, X. Liu, S. Neuendorffer, and H. Zheng, "Hyvisual: A hybrid system visual modeler," University of California, Berkeley, CA, Tech. Rep. Technical Memorandum UCB/ERL M03/1, Jan. 2003.