Preprints of the 2nd IFAC Conf. on Analysis and Design of Hybrid Systems (Alghero, Italy), 7-9 June 2006

BUILDING EFFICIENT SIMULATIONS FROM HYBRID BOND GRAPH MODELS

Christopher D. Beers * **Eric-Jan Manders** * **Gautam Biswas** *
Pieter J. Mosterman **

* *Dept. of EECS and ISIS, Vanderbilt University, Nashville, TN*
** *The MathWorks, Natick, MA*

Abstract: Embedded systems and their corresponding hybrid models are pervasive in engineering applications, therefore, systematic mathematical analysis using these models has become an important research area. Our approach to hybrid modeling with Hybrid Bond Graphs allows for seamless integration of physical system principles with discrete computational structures, but simulating the hybrid behaviors can be difficult and computationally expensive. In this paper, we develop a methodology that transforms Hybrid Bond Graphs into computational block diagrams and incrementally modifies the block diagram when mode changes occur. This forms the basis for a computationally efficient hybrid simulation algorithm. *Copyright © 2006 IFAC*

Keywords: Hybrid Bond Graphs, Causality, Hybrid Systems, Simulation

1. INTRODUCTION

Modeling and simulation play a central role in the design and operation of modern engineering systems that are made up of a large number of interacting components. Many of these systems are *hybrid*, i.e., they combine continuous and discrete behaviors. Hybrid simulation schemes must correctly handle 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].

A particularly intuitive physics-based modeling paradigm are Bond Graphs (BGs) [9]. They provide a uniform lumped parameter, energy-based topological framework across multiple physical domains (e.g., electrical, fluid, mechanical, and thermal). Hybrid Bond graphs (HBGs) extend the BGs by incorporating local switching functions that enable the reconfiguration of energy flow paths in the model ([15]). This allows for seamless integration of energetic modeling and model reconfiguration to handle hybrid behaviors. The inherent causal structure in BG models provides the basis for efficient conversion of BGs to computation models (e.g., [3, 9]). For HBGs, the computation model is more complex because junction switching during behavior generation results in dynamic updating of the causal assignments and the computational structure during execution. This paper presents an efficient method for constructing a simulation model for HBGs. Specifically, we create block diagram models, where run time changes in model configuration are handled by reconfiguring the data flow through the blocks of the model. We demonstrate the technique by creating a computational model in a commercially available simulation environment.

The choice of block diagram models is motivated by our work on simulation testbeds for fault detection and isolation and integrated systems health monitoring. For these applications, the simulation environment has to be designed in a way that component parameters in the model can be accessed and changed at run time to emulate degradations and faults in the system. The change profile for the parameters can take on a number of time-varying forms. In our work, the

component parameters are in 1-1 correspondence with BG parameters. and they map directly to individual blocks in the BG model.

2. HYBRID BOND GRAPHS (HBG) OVERVIEW

BGs are topological models that represent energy exchange pathways in physical processes [9]. The generic elements in BGs are energy storage, dissipative, transformation, and input-output elements. The dynamics of physical system behavior is captured by the transfer of energy between the components via the connected bonds and two idealized connections, called the 0- (common effort) and the 1- (common flow) junctions.

Introducing discrete behavior into continuous BGs has been investigated by several researchers [4, 11, 19]. HBGs introduce discrete changes in system configuration as idealized *controlled junctions* with two states *on* and *off* [15]. A Finite State Machine (FSM) implements the junction *control specification*. When the controlled junction is on, it behaves like a conventional junction. In the off state, all bonds incident on the junction are de-activated by enforcing a 0 effort or flow at the junction. Fig. 1 illustrates the discrete behavior of a controlled 1-junction. The system mode at any time is determined by a parallel composition of modes of the individual switched junctions.

To illustrate the concepts developed in this paper, we will use the bond graph model illustrated in Fig. 2. The system has a single source of effort, $v(t)$, that can be switched on or off, two capacities, C1 and C2, two inertias, L1 and L2, and two dissipators, R1 and R2. The switching junctions in the HBG (1_a and 1_b) are indexed with a subscript, and have a FSM (with the same subscript) attached with a dotted line. Two-port TF- and GY-elements are not included in our model, but they do not explicitly influence the causality in the system.

Computation and analysis with BG models is facilitated by the determination of causality, i.e., the input/output relationship between the effort and flow variables corresponding to BG elements. A standard algorithm for assigning causality to bonds is the Sequential Causality Assignment Procedure (SCAP) [9]. Modified causality assignment algorithms have been developed in other work [19].

Fig. 2 shows the causality information for configuration with both junctions on. All of the components

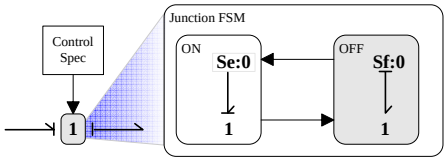

Fig. 1. Controlled junction as a Finite State Machine.

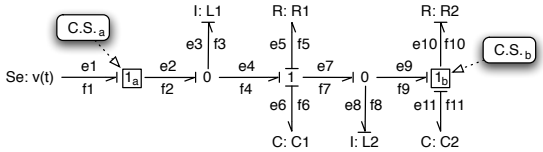

Fig. 2. Example system Hybrid Bond Graph.

in this model are in *integral causality*, which means that the constituent relations of the energy storage elements is formulated in their integral form (as opposed to the derivative form). In this paper, we make the assumption that all components will be in integral causality, and issues in dealing with derivative causality are discussed in section 5.

3. HBG COMPUTATIONAL MODEL

Computation models for continuous dynamic systems are represented by ordinary differential equations (ODEs), differential algebraic equations (DAEs), block diagrams (transfer functions), and signal flow graphs. In this work for the reasons described earlier, we adopt the block diagram representation.

3.1 Creating a causal model: Block Diagrams

We introduce the notion of a *determining bond* associated with every active (i.e., on) HBG junction. By definition, every active 0- (1-) junction in a valid bond graph will have one bond that determines the value of the effort (flow) for that junction. We label that bond as the determining bond for the particular junction. All other bond effort (flow) values are dependent and set equal to this effort (flow) value. Similarly, the flow (effort) value on a determining effort (flow) bond is an algebraic sum of the flow (effort) values of the other bonds that are connected to this 0- (1-) junction.

The determining bond plays a crucial role in mapping a HBG to a block diagram structure. Fig. 3 illustrates the mapping of a BG junction to a computational structure. The determining bond on the 1-junction, labeled with a 1, sets the flow at the junction (i.e., f2 = f3 = f1), and computes the dependent effort from the independent ones (i.e., e1 = - e2 - e3). The inputs to the adder blocks must carry appropriate signs depending on the orientation of the bond. In general, the algebraic constraints at a junction are of two forms, an equality constraint, represented by a dot, and a summation constraint, represented by an adder block. These relations are illustrated for both 0- and 1-junctions in Fig. 3.

Converting a BG model to a block diagram is a straightforward procedure when there are no algebraic loops and no elements are in derivative causality. First, each bond is replaced by two links, i.e., the effort and flow variables for the bond. Next, each junction is

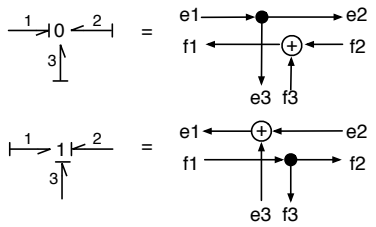

Fig. 3. Computational structures for bond graph junctions.

replaced by the algebraic constraints they impose (see Fig. 3). 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. The 1-1 mapping from bond graph elements to corresponding block diagram fragment can be found in most bond graph texts (e.g., see [9]).

The determining bonds establish the independent effort (flow) variables and the form of the algebraic equation for the corresponding flow (effort) variables at 0- (1-junctions). Therefore, the determining bond sets the block diagram structure, i.e., the direction of flows and efforts through the blocks. In our example, the determining bond for the leftmost 1-junction is bond 2, therefore, f2 is the driving flow, and it establishes f1 = f2. In addition, this also sets e2 = e1. The rest of the link and junction connections can be derived in a similar manner. Fig. 4 shows the resulting block diagram for our example system.

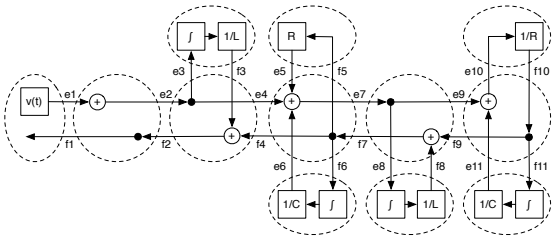

Fig. 4. Example HBG to block diagram conversion with both junctions active. The dotted circles encompass the original bond graph junctions and elements.

3.2 Efficient model transformation

For HBGs, the block diagram structure must handle junction switching. This is realized functionality as a control flow graph that dynamically reconfigures the computational block diagram when mode switches occur. If derivative causality was allowed, additional computational structures would be needed to update the system state at mode transitions.

Given a HBG with n switching junctions, there are potentially 2^n unique possible switching junction configurations. Pre-enumeration of all block diagram configurations offline and then selecting the appropriate one at run-time when junction configurations change is exponential in the number of switching junctions, and clearly a waste of space. On-line construction of the complete block diagram after each junction switch is space-efficient but wasteful in terms of computation time. Our solution to this problem is to construct a structurally adaptable block diagram model, and update the data flow paths through this model to match the causal structure when a junction switch occurs. Since we make the assumption that the system remains in integral causality, we exploit the locality principle for the propagation of causality changes through the model. This scheme may be combined with a caching mechanism that avoids having to recalculate causal assignment updates for discrete modes that have occurred previously.

3.3 Run-time operation

When junction switches occur in a HBG model the following changes are made to the existing block diagram to generate the block diagram for the new mode.

(1) Update the active HBG structure based on the junctions that change state. This procedure is described in Section 2.
(2) Evaluate the changes in the determining bonds for the junctions in the HBG structure, and propagate these changes to derive the block diagram structure for the new mode.

We illustrate the structural changes that result when the determining bond changes at a junction using our example. Consider the 1_a junction switching from on to off (Fig. 5). This event requires a new determining bond for the neighboring 0-junction. Since we assume preferred causality assignments, the I-element on this junction cannot switch its causal stroke, therefore, bond 4 must become the determining bond. This propagates to the adjacent 1-junction, and further until the 1-junction labeled 1_b, where the R-element absorbs the change by switching its causality assignment. Fig. 6 shows the resulting block diagram after the mode switch.

If the determining bond at a switching junction does not change, the effects of the mode switch do not propagate to adjacent junctions. For example, when the 1-junction labeled 1_b is switched off the determining bond for the adjacent 0-junction does not change. Consequently, there are no additional changes to the block diagram.

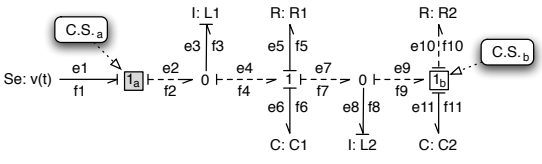

Fig. 5. HBG for the example system with junction 1_a off.

These examples illustrating the different effects of mode changes on determining bonds, indicate that an efficient scheme for incremental block diagram updating can be developed. The following lemmas make this explicit.

Lemma 1. Consider a junction switching from on to off or off to on. If the switch does not cause any change in the determining bond at its adjacent junctions, the only change in the junction structure of the block diagram is within the switching junction itself.

Lemma 2. Consider a junction switching from on to off or off to on. If the switch changes the determining bond at an adjacent junction, the change must be propagated, and the block diagram must be updated correspondingly.

Lemma 3. Consider two or more junctions switching simultaneously. Valid causal assignments are obtained regardless of the order in which the switches are evaluated.

The proofs for these lemmas are easily derived. Exploiting the fact that causality assignments determine the signal flow for the block diagram model, and that changes in causality have corresponding changes in the block diagram, we can minimize the computations required to generate an updated block diagram after a mode switch by identifying the points at which these will cause changes in the block diagram configuration, and then propagating these changes locally. This contrasts with the approach where one would rebuild the block diagram structure from scratch when mode changes occur. A general algorithm for our incremental updating procedure is described next.

A mode change is described by one or more junctions switching their state. When a 0- (1−) junction is deactivated, it provides a zero effort (flow) source with the causal strokes on each of the bonds away from (toward) the junction. When a junction is reactivated, it functions as a normal junction and its determining bond is updated. When a junction switches on or off, there are two possibilities: (i) the junction switching its state does not change the determining bond at any of its adjacent junctions, (ii) the switch changes the determining bond at an adjacent junction (Algorithm 1). For case (ii), if the adjacent junction contains a R-element and it's causality may be flipped to remove an extra determining bond or create a new determining bond, the propagation stops. Otherwise, the bond connected to another adjacent junction has to be flipped, and the propagation continues. Algorithm 2 describes the iterative causality propagation scheme that calls the *JunctionUpdate* algorithm. An initial causal assignment is generated before entering the mode switch event loop.

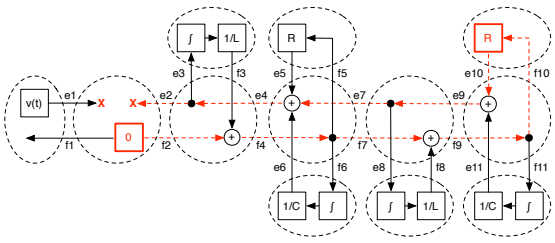

Fig. 6. Block diagram for the example system with junction 1_a off.

Algorithm 1 JunctionUpdate

```
if InvalidCausality then
  if NumDeterminingBonds=2 then
    if DeterminingBond=R-element then
      Change the R-element's causality
    else
      Change causality of an unchanged adjacent junction
      Call JunctionUpdate on adjacent junction
    end if
  else
    if R-element at junction then
      Change the R-element's causality
    else
      Change the causality of a previously unchanged adjacent junction
    end if
  end if
end if
```

Algorithm 2 Causality Propagation

```
Initial Causality Assignment
Find determining bonds
loop
  if ModeSwitch then
    Update causality of bonds at the switching junction
    for all causality changes at adjacent junctions do
      Call JunctionUpdate on adjacent junction(s)
    end for
  end if
end loop
```

4. IMPLEMENTATION

For this work we implement the block diagram simulation models using Simulink®[13]. The Simulink environment provides all the primitives to implement the block diagram structure for a bond graph, and the bond graph elements. For hybrid junctions, we must implement the control structure as well as the dataflow structure. Rather than implementing a switching junction using discrete Simulink blocks, or using Stateflow extensions to Simulink, we implement the switching junction as custom written S-functions in C/C++. The S-function implements the dataflow machinery for the junction, as well as the evaluation

of the control specification for the junction. For each bond connected to the junction, the S-function adds an input/output signal pair. The mapping of these signals to the effort/flow variables is determined dynamically by Algorithms 1 and 2. Note that we rely directly on the capabilities of the Simulink environment to detect the zero crossings, which define the mode changes.

An S-function obtains its current determining bond from a global data structure, which is updated at each junction switch. This data structure is updated by the control structure, represented explicitly in the Simulink model as the *CausalityUpdate* block. The *CausalityUpdate* block, when triggered by a junction switch, executes the *CausalityPropagation* algorithm that may reconfigure the junction input-output relations by updating the global data structure. Thus the physical structure of the Simulink model remains static, and all dynamic updates to the data flow through the Simulink model are handled by the C/C++ code that implements the controlled junctions.

The simulation model is created programmatically using the process described above through the Simulink Application Programming Interface. In other work, we have developed a physical system modeling environment that supports the building of HBG models [12]. An automatic model translation procedure creates the builder network that is then passed to the Simulink code generator. The Simulink model for the example (Fig. 2) is shown in Fig. 7.

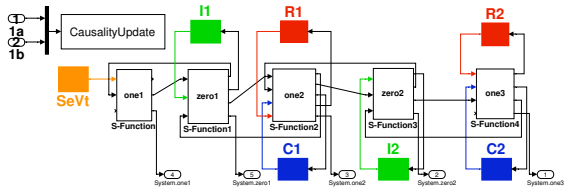

Fig. 7. Example system generated Simulink model.

The simulation results for the example system are shown in Fig. 8. All four configurations of the bond graph are visited, and the efforts and flows are graphed, along with the switching junction state.

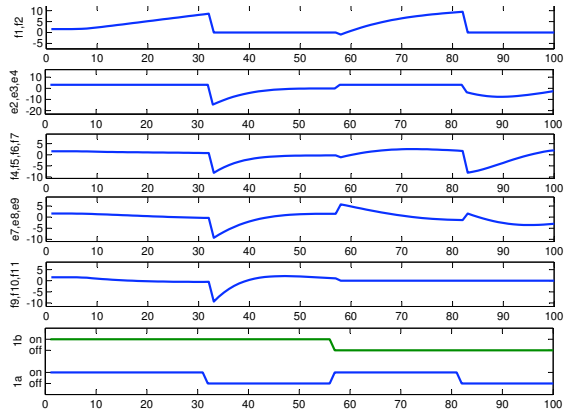

Fig. 8. Simulation outputs for a simple control sequence through each discrete state.

5. RELATED WORK

The concept of a determining bond introduced in this paper builds up on the methods presented in [9] on the efficient derivation of analytic and simulation models from BGs by exploiting the inherent causality relations in the underlying BG model. Extensions to hybrid systems have been studied by a number of researchers (e.g., [5, 2, 17, 10]), but few have discussed efficient methods for reassigning causality and regenerating the computational model at runtime when junction switches occur. As we have discussed earlier, our focus is on block-diagram based simulation models to facilitate component-oriented diagnostic analysis of systems [14]. [18] have looked at the notion of switching power junctions, and the explicit switching of effort-deciding or flow-deciding bonds when model configurations change. They define the notion of causality invariance and apply their method to special classes of electrical systems. The approach presented in this paper is more general, and the algorithms we have developed apply to any HBG model where the system remains in integral causality for all modes of operation. Similar work on switching power systems is reported in [7, 11], but both papers focus on equation generation based on Hamiltonian methods, which is somewhat tangential to our approach on block diagram-based simulation models. Buisson, et al. [4] use the notion of switched bond graphs, a very similar approach to our HBG models. But unlike the approach described in this paper, they do not use an incremental approach to causality re-assignment when junction switches occur. Their focus too is equation generation when mode changes occur using implicit state equation methods. [6] deal with switching junctions and causality re-propagation in hybrid systems, but their focus has mainly been on situations where energy-storage elements switch into derivative causality. The methods presented in this paper do not deal with situations where the system model goes into derivative causality.

HyBrSim is an experimental application for HBG modeling and simulation ([16]). The simulation algorithm includes mechanisms for performing event detection and location based on a bisectional search, and the algorithm can handle runtime causality changes (including derivative causality) when junctions switch on and off. HyBrSim runs in interpreted mode and 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.

A very different simulation implementation can be taken as well, where the causal changes are handled based on the use of an implicit formulation of the junction equations ([16]). This eliminates the difficulty of explicitly handling causal changes during runtime. Instead the complexity is handled by the implicit equation solver, which typically relies on an algebraic

equations solver. However, it is not clear how this method scales up for larger models, and when multiple junctions switch simultaneously.

6. CONCLUSIONS

The work presented in this paper uses physical system modeling semantics as defined by BGs and HBGs to impose semantic structure on hybrid computational models in Simulink. Other elegant computational approaches, such as Ptolemy and HyVisual [8] 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, and propose a more formal model of computation that is linked to physical principles.

ACKNOWLEDGMENTS

This work is supported in part by grants from NSF ITR grant number: CCR-0225610, NSF SGER-052067, and NASA ALS contract number: NCC 9-159.

REFERENCES

[1] P. Antsaklis. A brief introduction to the theory and applications of hybrid systems. *Proc IEEE*, 88(7):879–887, 2000.

[2] W. Borutzky, J.F. Broenink, and K.C.J. Wijbrans. Graphical description of physical system models containing discontinuities. In A. Pave, editor, *Modelling and Simulation, Proc. of the European Simulation Multiconference*, pages 208–214, Lyon, France, June 1993.

[3] Jan F. Broenink. 20-sim software for hierarchical bond-graph block-diagram models. *Simulation Practice and Theory 7*, 7(5–6):481–492, 1999.

[4] 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*, pages 47–63, 2002.

[5] 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*, 337:579–600, 2000.

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

[7] G. Escobar, A. van der Schaft, and R. Ortega. A hamiltonian viewpoint in modeling of switching power converters. *Automatica*, 1999.

[8] Christopher Hylands, Edward A. Lee, Jiu Liu, Xiaojun Liu, Stephen Neuendorffer, and Haiyang Zheng. Hyvisual: A hybrid system visual modeler. Technical Report Technical Memorandum UCB/ERL M03/1, University of California, Berkeley, CA, January 2003.

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

[10] F. Lorenz and H. Haffaf. Combinations of discontinuities in bond graphs. In *Proc. Intl. Conf Bond Graph Modeling Simulation*, pages 56–64, Las Vegas, NV, January 1995.

[11] 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.

[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, March 2006. IEEE CS Press.

[13] MathWorks. MATLAB®/Simulink® The MathWorks, Inc, Natick, MA. (http://www.mathworks.com/products/simulink/).

[14] P. J. Mosterman and G. Biswas. Diagnosis of continuous valued systems in transient operating regions. *IEEE Trans. Syst., Man Cybern. A*, 29 (6):554–565, 1999.

[15] P. J. Mosterman and G. Biswas. A theory of discontinuities in physical system models. *J Franklin Institute*, 335B(3):401–439, 1998.

[16] P.J Mosterman. Hybrsim - a modeling and simulation environment for hybrid bond graphs. *Journal of Systems and Control Engineering - Part I*, 216, 1:35–46, 2002.

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

[18] Amod C. Umarikar and L. Umanand. Modelling of switching systems in bond graphs using the concept of switched power junctions. *J Franklin Institute*, 342:131–147, 2005.

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