Proceedings of the 2005 Winter Simulation Conference
M. E. Kuhl, N. M. Steiger, B. F. Armstrong, J. E. Joines, eds.

HYBRID DISCRETE EVENT SIMULATION WITH MODEL PREDICTIVE CONTROL FOR SEMICONDUCTOR SUPPLY-CHAIN MANUFACTURING

Hessam S. Sarjoughian
Dongping Huang
Gary W. Godding

Arizona Center for Integrative Modeling & Simulation
Computer Science & Engineering Dept.
Arizona State University
Tempe, AZ 85287-2803 U.S.A.

Wenlin Wang
Daniel E. Rivera

Control Systems Engineering Laboratory
Chemical & Materials Engineering Dept.
Arizona State University
Tempe, AZ 85287-2803 U.S.A.

Karl G. Kempf

Decision Technologies
Intel Corporation
Chandler, AZ 85226 U.S.A.

Hans D. Mittelmann

Mathematics & Statistic Dept.
Arizona State University
Tempe, AZ 85287-2803 U.S.A.

ABSTRACT

Simulation modeling combined with decision control can offer important benefits for analysis, design, and operation of semiconductor supply-chain network systems. Detailed simulation of physical processes provides information for its controller to account for (expected) stochasticity present in the manufacturing processes. In turn, the controller can provide (near) optimal decisions for the operation of the processes and thus handle uncertainty in customer demands. In this paper, we describe an environment that synthesizes Discrete-EVent System specification (DEVS) with Model Predictive Control (MPC) paradigms using a Knowledge Interchange Broker (KIB). This environment uses the KIB to compose discrete event simulation and model predictive control models. This approach to composability affords flexibility for studying semiconductor supply-chain manufacturing at varying levels of detail. We describe a hybrid DEVS/MPC environments via a knowledge interchange broker. We conclude with a comparison of this work with another that employs the Simulink/MATLAB environment.

1 INTRODUCTION

In the semiconductor supply-chain, models of decision planning and manufacturing processes are essential to deal with frequently changing customer demands while remaining financially competitive (Kempf 2004). While there exist well-known approaches to model complex decision and process models independently, modeling of their interactions depends mostly on customized abstractions viewed from decision models or simulation models. For example, we can consider discrete event and model predictive control to represent process dynamics and decision controls that can manage material flow of a semiconductor supply-chain, respectively. Discrete Event Simulation (DES) is generally considered suitable for modeling and simulating the operations (processes) of semiconductor supply-chain network systems. Likewise, a variety of optimization algorithms are most suitable for planning. An attractive approach for planning is known as Model Predictive Control (MPC) (Wang, Rivera and Kempf 2005). Unlike Linear or Integer Programming which is considered for strategic control of supply-chain processes, MPC is aimed at tactical control. Tactical control is concerned with short term (daily to several weeks) decision making whereas strategic control focuses on long term decisions (few to several months).

Use of different modeling methodologies requires integrating the process dynamics and decision planning. Such integration could be specified at different levels of abstraction. Integration of models could be via low-level programming, high-level interoperability techniques, or multi-model composability approaches. The basic concept and approach for Knowledge Interchange Broker (KIB) was proposed to support multi-formalism model composition [i.e., discrete-event and agent models (Sarjoughian and Plummer 2002) and more recently has been used for composing discrete-event and linear program models for semi-

conductor supply-chain networks (Godding, Sarjoughian and Kempf 2004)]. The KIB prescribes data and control mapping, synchronization, concurrency and timing across composed formalisms. Given these basic capabilities, composition of different modeling formalisms are specified in terms of generalized and domain-specific syntax and semantics.

In this paper we describe a prototype environment where a discrete event semiconductor process model is composed with a model predictive control decision model. In Section 2, we highlight closely related work in modeling and simulation of semiconductor supply-chain networks considered in this paper. In Sections 3 and 4, we describe the DES and MPC models that are to be composed. In Section 5, we describe a general approach to composing DES and MPC modeling approaches. A prototyped environment integrating the DEVSJAVA and MATLAB environments is described in Section 6. Conclusions and future work are discussed in Section 7.

2 RELATED WORK

Numerous articles have been devoted to the study of combining manufacturing processes and decision control models for the domain of semiconductor supply-chain networks. A common approach is to develop customized code to integrate different simulation and planning environments. These customizations may use a variety of software engineering techniques ranging from low-level programming to middleware technologies where interoperability concerns between disparate implementation can be systematically accounted for. These approaches primarily rely on interoperability to handle model composability.

Recently an agent framework has been proposed as a common basis to model and integrate different parts of supply-chains. In particular, a multi-agent framework has been developed where physical and decision models are integrated using a library of common elements of a supply-chain such as factories and control policies, and other elements that support their interactions (Swaminathan, Smith and Sadeh 1998). An alternative approach has been developed using the concept of Knowledge Interchange Broker to compose Discrete-Event Simulation (DES) and Linear Programming (LP) optimization models (Godding, Sarjoughian and Kempf 2004). In this work, appropriate data transformations and control have been developed for models that can be described in the DEVS and LP formalisms.

To our knowledge, no modeling and simulation environment can support hybrid discrete-event modeling and model predictive control in general and in particular for semiconductor supply-chain networks. Instead, there exist approaches where MPC is used with continuous and discrete-time models. For example, the commonly used Simulink/MATLAB environment has been used to develop a discrete-time simulation model of the discrete event manufacturing processes with an MPC consisting of a linear, time-invariant process model and a quadratic programming optimization model (Wang, Rivera and Kempf 2005).

This Simulink/MATLAB block diagram discrete-time modeling engine provides connectivity to the MPC tool box. Synchronous interactions between discrete-time and MPC models are supported for primitive data types, although given the knowledge for the built-in Simulink/MATALB data and control schemes, customized exchange may be developed. These customized interactions, however, seem difficult to support in a generic setting and thus rely on ad-hoc customization in developing process and MPC models and, more importantly, their combination. This observation can be made about other environments where a core modeling and simulation paradigm is extended with others using a combination of interoperability and software engineering techniques.

Finally, in recent years, model composability has attracted researchers and increasingly turned their attention to developing concepts and techniques to tackle inherent complexities associated with component-based model specification. The main goal of these efforts is to develop basic theories and techniques to tackle dissimilarities of models to be synthesized using model abstractions, metamodeling, and model transformation [for examples see (Sarjoughian and Cellier 2001; Mosterman and Vangheluwe 2002; Davis and Anderson 2004)].

3 PHYSICAL PROCESS MODELING

Common physical models of manufacturing a supply-chain network consists of four types of nodes (models). These are inventory, factory, shipping link, and customer. These entities have common structures and behaviors. It is, therefore, important to develop a common interface specification for these nodes. Each node type must be able to receive materials or products (data) and accept decision commands (control). Specific functions need to be specified for each node type. Generally material flow is assumed to be uni-directional (feed-forward) from the supplier to final customers. Control flow, however, may include feedback.

Release (or receipt) of materials from inventory (or other nodes) can be characterized in terms of (i) quantity, (ii) type, (iii) time, and (iv) destination. *Quantity* refers to the number of (specific) items to be built and/or sent out. *Type* distinguishes among different kinds of materials to be sent out. *Time* refers to a specific time instance for release of materials. *Destination* refers to nodes that are to receive the released materials.

The inventory model has *capacity* and *delay*. The states of the inventory model (e.g., Die and Package) include inventory levels (possibly stochastic) for each prod-

uct it can hold. Inventory stores materials at the time they are received and releases them as it is instructed.

The factory model represents manufacturing, assembly, test, and finish processes, or some combination such as assembly and test (see Figure 1). It can have *capacity*, *throughput time* (*TPT*), and *yield*. The actual values of TPT and yield are generally stochastic; it may depend on the current load. The factory node can change, assemble, and split products. With the change operation, one input product is made into another product. For example, as shown in Figure 1, raw silicon wafers are fabricated into die and then tested in Fab/Test1. The assembly operation represents two or more products that are assembled into one product. In the Assembly/Test2, one package must combine with one die. There must exist enough packages and dies to begin their assembly into semi-finished goods. The split operation involves the manufacturing of two products with different properties. For example, the items coming into Assembly/Test2 may be stochastically split into two bins, one of which contains high-speed devices while the other contains low-speed devices. This kind of separation can also be modeled as two separate inventories. The assembly operation is generally controlled externally (via decision models), while the split operation is a characterization of the factory itself.

The shipping node models transportation delay. It can be thought of as a specialized factory model where it cannot change products. The customer node models customer behavior. It can send product demands to both the decision system and process system (e.g., customer warehouse) for decision making and order processing respectively. Two types of customer demands may be modeled: forecasting demands and actual demands.

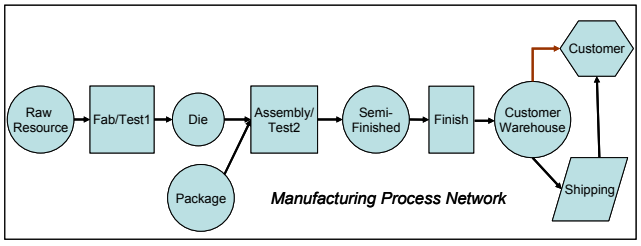

Figure 1: Semiconductor Manufacturing Process Network

Ports can be used to represent input/output interfaces for all network nodes. The ports are distinguishable as *data* and *control* input and output ports (Singh, Sarjoughian and Godding 2004). A node may have multiple data ports to support sending different products to multiple destinations. The decisions of how much (quantity), what (material type), time (when to send), and destination are determined based on external information that is local to the process model. For example, where to send material is determined by a decision model and product split is determined by the manufacturing node.

External and local controls are defined for the supply-chain network nodes. The external control represents the control commands from the decision model to the supply-chain network nodes. For example, the release commands for each inventory can be determined using an optimization scheme given the current inventory levels, factory work-in-progress (WIP), customer demands and some constraints among them. Some decisions, however, may be formulated locally. For example, an inventory may release a quantity of products given the external control release command. However, an inventory's external control release may be constrained given the maximum capacity of the receiving (downstream) node and its available inventory level. Such a local control policy is shown in Figure 2.

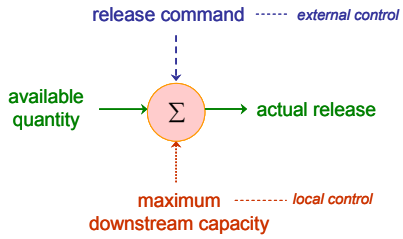

Figure 2: Local Control Policy for Inventory

3.1 DEVS Models

The above manufacturing process network dynamics are modeled and simulated using the Discrete Event System Specification (Zeigler, Praehofer and Kim 2000). This approach supports developing models with well-defined structures and behaviors. It offers a framework for modeling and simulating discrete or continuous systems as atomic and hierarchical coupled models. The simulation of models is based on a simulation protocol that ensures correct execution of the models—i.e., enforcing causality, concurrency, and timing among atomic and coupled models.

Atomic models have input/output ports and values. The behavior of the atomic model is specified in term of state variables and functions. A model can have autonomous and reactive behavior specified in terms of *internal transition* and *external transition* functions. *Output* function allows the model to send out messages. *Time advanced* function captures timing of models. *Confluent* function can be used for modeling simultaneous internal events and external events. Every coupled model has well-defined interfaces, as in atomic models, and consists of one or more atomic model or coupled models. Complex models can be hierarchically constructed from these two types of models.

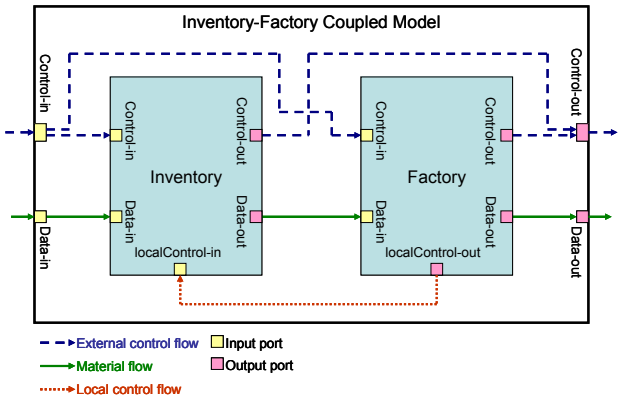

Figure 3: Structural Composition of Inventory and Factory Models

The supply-chain network nodes lend themselves to be specified in terms of atomic and coupled models. These models can be interconnected to form an arbitrary supply-chain network using coupling topologies. Figure 3 shows a coupled model which includes one atomic inventory model and one atomic factory model where product and local and external controls are separately modeled.

DEVSJAVA is a software environment that was developed based on the DEVS framework and implemented in the JAVA programming language (DEVSJAVA 2002). It provides a component-based (object-oriented) engine for atomic and coupled simulation modeling. The combination of component-based modeling and software implementation of DEVSJAVA provides flexibility and scalability to develop the supply-chain network processing models.

4 MODEL PREDCITIVE CONTROL MODEL

Model Predictive Control (MPC) is commonly used for control of highly stochastic processes where selection of control actions, based on optimization, is desired. The importance of MPC compared with traditional approaches is due to its suitability for large multi-variable systems, handling of constraints placed on system input and output variables, and its relative ease-of-use and applicability. In MPC, current and historical measurements of a process are used to predict its behavior for future time instances. A control-relevant objective function is optimized for calculating a sequence of future control commands that can satisfy some predefined system constraints. For details of the objective function, refer to (Wang, Rivera and Kempf 2005).

It has been shown that MPC can be used effectively as a tactical controller for high volume supply-chain semiconductor networks having capacitated, nonlinear, and stochastic demands (Wang, Rivera and Kempf 2005). An MPC design for semiconductor supply-chain is shown in Figure 4. It consists of a System Prediction Model and an Optimizer. The real system has been modeled using DEVSJAVA. In this paper, the real-system is replaced with a DEVS simulation model. In this study, the operation of MPC can be described informally as follows. The real system (i.e., simulatable DEVS process model) sends its current outputs (e.g., semi-finished goods inventory level) to the system prediction model given measured disturbances (i.e., actual customer demand). The system prediction model then computes future outputs (i.e., controlled outputs) for some finite number of time steps. The error between future outputs and target trajectories (i.e., expected customer demand) is sent to the optimizer where optimized control outputs (referred to as manipulated variables) are calculated based on some constraints and objective functions over some time horizon—i.e., moving horizon (for manipulated variables) and prediction horizon (for controlled variables). This optimization will be repeated using the receding horizon concept once the new information is available. In addition, the MPC has a filter gain that can respond quickly to inevitable signal to noise ratio changes while avoiding undesirable oscillatory control regimes. The predictive control for the first time step is sent to simulated system as well as the system prediction model. The above steps are repeated using the updated simulated system states and disturbances for a desired simulation period.

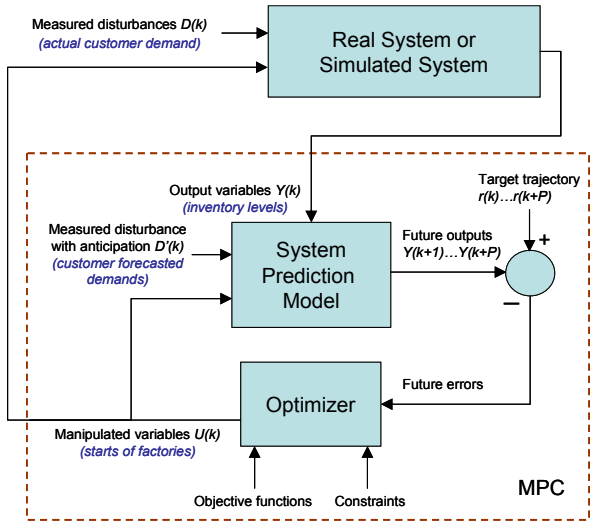

Figure 4: Canonical MPC Model

The MPC is supported by commercial tools such as MATLAB (Mathworks 2002). In our work, the MATLAB's QP solver is replaced with the more efficient and robust MATLAB-QP version of the interior point NLP code LOQO (Vanderbei 1999). MATLAB supports a graphical user interface for modeling and observing simulation results.

5 COMPOSING DEVS AND MPC USING KIB

The key concept behind Knowledge Interchange Broker is the ability to compose models even though each model has its own distinct syntax and semantics. Clearly, not only must each model be executed using its own well-defined protocol, but their interoperation must also be guaranteed correct. In the case of DEVS and MPC, the KIB's execution engine is designed to ensure that the DEVSJAVA simulation and the MATLAB solver interact correctly. Next, we detail the generic KIB approach for composing and interoperating the classes of DEVS and MPC modeling and simulation paradigms.

For tactical control of a semiconductor supply-chain, we consider composing a discrete event simulation (i.e., DEVS) and a tactical controller (i.e., MPC) for the simple example shown in Figure 4. The system prediction model of the DEVS Process Model used within the MPC is modeled as discrete linear and time-invariant model (see Figure 4). The system predictive model is described as in equation (1).

$$\begin{aligned} X_{k+1} &= AX_k + B_u U_k + B_v D_k \\ Y_{k+1} &= CX_k \end{aligned} \tag{1}$$

where $U_k \in \Re^{m_u}$ is a vector of manipulated variables, $X_k \in \Re^{m_x}$ is a vector of state variables, $Y_k \in \Re^{m_y}$ is a vector of output (or controlled) variables, and $D_k \in \Re^{m_v}$ is a vector of measured disturbance variables.

As a tactical controller, the MPC manipulates the starts of the factories to satisfy the forecasted customer demands (D'_k) given the actual customer demands (D_k) while maintaining the inventories at their desired levels. The controlled variables are the inventory levels; the manipulated variables are starts of each factory; the customer demands are treated as measured disturbance variables with anticipations. We note that the MPC does not receive the actual releases (see Figure 2) from the manufacturing process model, although a different formulation of the MPC may.

The MPC manipulates the start of the factories of the semiconductor process model as follows. We have simplified the process model to have one product.

1. At the initialization, the inventory set-point trajectories are specified. The model attributes such as average *TPT* and *yield* for each factory model are set. The distribution of some stochastic behaviors, such as distribution of the *TPT* and *yield*, are also set at initialization.
2. At time interval *k*, the MPC receives the current inventory levels (Y_k). It also receives forecasted customer demands (D'_k). The system prediction model has the previous inventory levels $Y_{k-1}, Y_{k-2}, \ldots$, the previous starts of each factory $U_{k-1}, \cdots, U_{k-m}$, and the previous customer demands $D_{k-1}, D_{k-2}, \ldots$. To calculate the next start for each factory, the controller operates in two phases:
   (a) Estimation. The controller uses all the past measurements, inputs, and the current controlled variables to calculate the inventory levels for a prediction horizon P (a finite integer ≥ 1), $Y_{k+1}, Y_{k+2}, \ldots, Y_{k+P}$.
   (b) Optimization. Values of the future inventory level trajectories, anticipated customer demands, and constraints are specified over a finite horizon of future sampling instants $k+1, k+2, \ldots, k+P$. By solving a constraint optimization problem, it computes the starts of each factory in the future horizon M ($P \geq \mathrm{M} \geq 1$), $U_k, U_{k+1}, \ldots, U_{k+M-1}$.
3. The starts at time k (U_k) are sent to the process simulation model. Each inventory model releases products to its downstream factory given its local control policy shown in Figure 2.
4. At the next time interval *k+1*, continue with step 2.

5.1 Structural Composition Specification

Specification of model compositions needs to support structural composition of DEVS and MPC models. As described earlier, (data and control) input and output ports are used as interface structures for DEVS models to interact with DEVS and non-DEVS model components. The interface structure of an (atomic or coupled) model consists of sets of input and output ports with values to be exchanged with model components. An example of structural specification of an inventory model is given as equation (2).

$$\begin{aligned} X &= \{(data-in, Lots), (control-in, Command), \\ &\quad (localControl-in, Capacity)\} \\ Y &= \{(data-out, Lots), (control-out, [BOH, Released])\} \end{aligned} \tag{2}$$

The set of inputs and outputs are *X* and, *Y* with *(port, message)* representing port name and message type. The *message* can be complex such as *Lot* which contains a collection of multiple product types with different cardinalities. In DEVSJAVA, each lot is modeled to have a default or user-specified size (e.g., 100 products per lot).

For the MPC model, its discrete-time system model (denoted as I_{10}, I_{20}, I_{30}, M_{10}, M_{20} and M_{30} in Figure 5) and optimizer do not use ports. Instead, the interface structural specification of MPC is "vectors of variables." The input to the controller from the simulation system is a vector of controlled variables and a vector of measured disturbance variables, while the output from the controller to the simulated system is a vector of manipulated variables (see Figure 5). Table 1 shows an example of input/output mapping between the DEVS and MPC models.

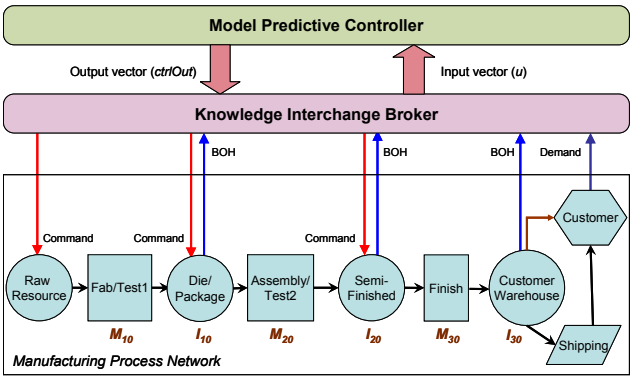

Figure 5: Composition of Manufacturing Process Network and MPC via KIB

The structural specification must provide well-defined structural information translation from the DEVS model to the MPC and vice versa. Clearly, the data types described in the two modeling formalisms are different. Messages used in the DEVS models are of type *Entity*, which is the base data structure used for building messages with arbitrary complexity. The MPC variable types, in contrast, generally have primitive types (e.g., reals), although complex variable types may be defined by users.

Table 1: Message Mapping for DEVS and MPC Models

DEVS Model	MPC Model
(Inventory, control-out, BOH)	y_i, a member of vector m_y controlled variables
(Inventory, control-in, command)	u_i, a member of vector m_u manipulated variables
(Customer, control-out, demand)	v_i, a member of vector m_v measured disturbance variables

The principles of knowledge reduction and augmentation are important for handling differences between two modeling formalisms. Knowledge reduction is generally simpler as we throw away information in the process of translating one message type to another. For example, the message Beginning On Hand (BOH) in the DEVS inventory model represents current inventory level. It may contain not only the product amount but also the product name. When the message is transformed to an MPC variable, the name may not be necessary. In contrast, knowledge augmentation is more difficult. For example, the manipulated variable sent from the MPC to the DEVS model represents a release command for some specific product from a specific inventory. How to augment (map) the MPC release command such that it becomes a DEVS message needs to be specified based on the MPC and DEVS formalisms (see Section 6 for details).

Knowledge reduction and augmentation can contain data (message) aggregation and disaggregation. The former refers to combining multiple data values (or a set of messages) into a single data value (a message) and the latter refers to the inverse. For example, an MPC data variable can be disaggregated so that it can be sent to multiple DEVS input ports.

5.2 Behavioral Composition Specification

Execution of composed DEVS and MPC models needs to be guaranteed correct. To satisfy correct execution of process and decision models, we need to devise synchronization and timing that conform to the DEVS simulation protocol and the MPC's simulation algorithm and solver.

The DEVS specification includes a set of states and transition functions. The state variables (e.g., average throughput time and average yield in the factory model) must be updated such that output to input causality and timing are satisfied. This requires ensuring the ordering of external, internal, and output function executions while interacting with the KIB (and therefore the MPC).

Likewise, the execution of the MPC must be ensured given its interaction with the KIB. The system prediction model has a homomorphic relation to the DEVS process mode. The optimizer is a set of constraints and objective function specification based on mass conservation relationships among the inventory, manufacturing, and transportation models. For example, the mass conservation relationships for Die/Package inventory level (I_{10}) and for Fab/Test1 WIP (M_{10}) can respectively be expressed as equation (3) (Wang, Rivera and Kempf 2005):

$$\begin{aligned} I_{10}(k+1) &= I_{10}(k) + Y_1 C_1(k-\theta_1) - C_2(k) \\ WIP_{10}(k+1) &= WIP_{10}(k) + C_1(k) - Y_1 C_1(k-\theta_1) \end{aligned} \quad (3)$$

The variables θ_1 and Y_1 represent the nominal throughput time and yield for the Fab/Test1 node, while C_1 and C_2 represent the daily starts that constitute inflow and outflow streams for I_{10} and M_{10}. Similar constraints can be written for other elements of the manufacturing process.

Consistent model attributes of the composed model—e.g., stochastic and non-stochastic *TPT* and *Yield* for the simulated (e.g., Fab/Test1) and predictive factory (e.g., M_{10}) models, respectively—are required. That is, the common attributes of the process and decision models must be kept consistent since they represent semantically the same knowledge across the composed model. These attributes, however, do not necessarily have to be identical, but they carry the same information at different levels of abstraction. The consistency of the model attributes and their exchanges must be ensured by the KIB both at initialization of and during the simulation.

Both DEVS and MPC models have well-defined timing properties. The former has a continuous-time base while the latter has a discrete-time base. Since the MPC models execute using time-stepping, the KIB's input and output events are synchronized using a discrete time clock. This is appro-

priate since the physical process model timing is a multiple of the decision model timing. For example, a process simulation execution cycle can be hourly, daily, or weekly. The decision planning can run at the same frequency as the process model simulation, faster, or slower. For example, the process model can be simulated using daily time-step while the planning can have a weekly time horizon.

From the MPC point of view, when it receives the process model states (i.e., controlled variables), the controller assumes the process model states remain static while it computes the values of the manipulated variables and gives control back to the process model—until it receives the process model states for the next time step. For the DEVS models, the timing period from sending out state information to receiving control messages is dependent on the frequency selected between the two models. If the process and decision models have the same frequency, the process model must receive control messages before it changes its states and starts a new cycle. If they run in different frequencies, the process model must receive the control messages at the correct time step.

For the DEVS/MPC composition, the KIB is designed to support synchronous control. In this mode, once the DEVS simulator sends outputs to the KIB, the simulation stops until it receives the MPC manipulated variables. This form of synchronization needs to only maintain a single logical clock for executing composed models. In the prototype described next, the KIB execution protocol synchronization is defined in terms of the DEVS simulation event ordering and clock.

6 PROTOTYPE KIB DESIGN

A prototype of the KIB targeted for composing DEVS and MPC models was designed and implemented using parts of two existing KIBs (Sarjoughian and Plummer 2002; Godding, Sarjoughian and Kempf 2004). The DEVS/MPC simplified architecture is shown in Figure 6. It consists of DEVSJAVA, KIB, and MATLAB.

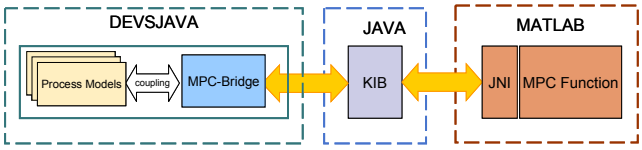

Figure 6: DEVS/MPC with KIB Conceptual Architecture

A software design and implementation of the KIB is shown in Figure 7. It includes *DEVSDecisionInterface, KIB Data Transformation Manager*, and *MPCInterface.* The DEVSDecisionInterface is the interface between DEVSJAVA and the KIB. The KIB Data Transform Manager handles data transformation (mappings). The MPCInterface is an interface between MATLAB and the KIB.

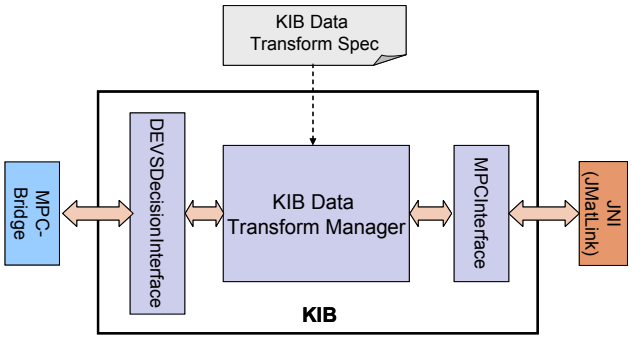

Figure 7: Software Components in KIB

6.1 DEVS Decision Interface and MPC-Bridge

The responsibility of this interface is to pass messages from the DEVS models to the KIB and vice versa. The messages include *initMessage* and *statusMessage* from DEVS to KIB and *controlMessage* from KIB to DEVS.

The process model can interact with the KIB via the DEVSDecisionInterface object method invocations or via ports (see Figure 8). If we choose method invocation, it turns out that the communication between the process model and the KIB is synchronous. If we use ports, we can take advantage of the DEVS simulation control protocol to handle timing properties for the composed DEVS and MPC model. Therefore, a special atomic model *MPC-Bridge* was developed as a proxy between the KIB and DEVS models. It collects state information from the process models through the input ports and then transfers them to the KIB.

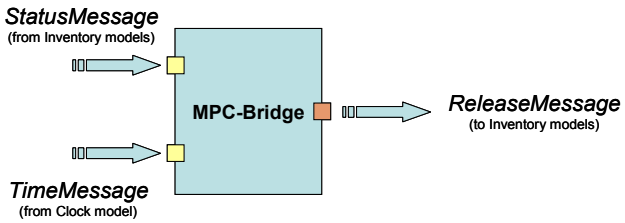

Figure 8: MPC-Bridge Structural Specification

Similarly, the MPC-Bridge as shown in Figure 8 receives the control messages (i.e., MPC manipulated variables) from the KIB which are then sent to the process models at appropriate time intervals. The interaction of the *MPC-Bridge* and the process models are through DEVS ports, while the interaction between the *MPC-Bridge* and the KIB uses method invocations. We note that the *MPC-Bridge* supports the KIB in two ways. First, it acts as a bridge between KIB and DEVS and second it provides message synchronization with MPC—data sent to the MPC model and commands sent to the DEVS model.

6.2 MPCInterface

This interface defines interactions between the KIB and the MPC which is specified as a MATLAB function. The MPC is defined as shown in equation (4).

$$\begin{aligned} &function\ ctrlOut = mpcFunction(time, u) \\ &\text{where } time \in \mathrm{N},\ u \in \Re^4,\ ctrlOut \in \Re^3 \end{aligned} \tag{4}$$

Here *time* is an input variable representing a discretized time index. The controlled input variable u is a vector. The manipulated variable *ctrlOut* is the output variable. Both u and *ctrlOut* variables are of the same type (i.e., double).

The functionality of the *MPCInterface* includes (a) loading the MPC function file to the appropriate internal MATLAB workspace, (b) transforming the MPC model in the KIB to the corresponding MATLAB function call, (c) making the MATLAB call through the JMatLink (Müller 2002) which is based on the Java JNI, and (d) transforming the output variables from MATLAB to the KIB.

As mentioned above, the MPC needs the real-system's (i.e., simulation model's) previous input, output, and state variables to calculate its next manipulated variables (see Figure 5). This previous information can be stored in MPC function space instead of the KIB. At initialization, the MATLAB engine creates a workspace for MATLAB functions, in which the previous information can be stored until it is closed. Therefore, given the MATLAB environment, it is not necessary to transfer its previous state information back and forth to the KIB.

6.3 KIB Data Transformation Manager

This main responsibility of the transformation manager is to coordinate interactions between the DEVSDecisionInterface and the MPCInterface given the defined data and control mappings and transformations. The main functionalities of the manager can be summarized as

- Maintain DEVS models and MPC function information and their composition specifications;
- Transform data information between the two modeling formalisms;
- Control the interactive execution between the two modeling formalisms.

The manager contains two types of data nodes: *DEVSModelNode* and *MPCFunction* to keep the DEVS and MPC function information, respectively. Each process model in DEVS has a corresponding *DEVSModelNode* in the KIB to store its information (e.g., status information). The *MPCFunction* consists of *MPCParameters*, each of which has well-defined mapping information to the *DEVSModelNode*. The mapping information corresponds to the KIB composition specified in XML, which is a standardized schema for data exchange.

A simplified structural composition specification is shown in Figure 9. The XML specification shows the DEVS output variables and the MATLAB input variables (see Equation (4)). The sequence diagram shown in Figure 10 illustrates the processing taking place inside the KIB including message mapping and transformation. It shows message passing and method invocations from the MPC-Bridge to the MPCInterface.

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- MATLAB function configuration for interacting with DEVS -->
<FUNCTION Name="mpcFunction">
   <!--type can be customized-->
   <INPUT Name="Time" Type="double" size="1">
      <VALUE>0</VALUE>
   </INPUT>
   <INPUT Name="u" Type="double" size="4">
      <VALUE Model="DiePackage" Product="x">0</VALUE>
      <VALUE Model="SFGI" Product="x">0</VALUE>
      <VALUE Model="CW" Product="x">0</VALUE>
      <VALUE Model="Customer" Product="x">0</VALUE>
   </INPUT>
   <OUTPUT Name="yOut" Type="double" size="4">
      <VALUE Model="RawI" Product="x" Destination="Fab"></VALUE>
      <VALUE Model="DiePackage " Product="x" Destination="AT"></VALUE>
      <VALUE Model="SFGI" Product="x" Destination="Finish"></VALUE>
      <VALUE Model="CW" Product="x" Destination="Shipping"></VALUE>
      </OUTPUT>
</FUNCTION>
```

Figure 9: KIB Composition Specification

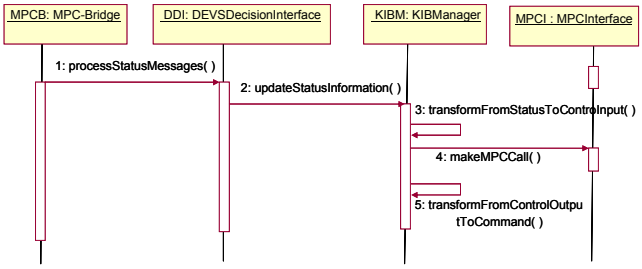

Figure 10: Data and Control Mapping Snippet

6.4 Simulation Test Case

The example described in earlier sections has been developed using the DEVS/MPC approach. The manufacturing network model components are described as atomic and coupled components which interact with the MPC model described above. The average throughput times and capacities [days, # of products] for Fab/Test1, Assembly/Test2, Finish, and Shipping are [35, 45000], [6, 7500], [2, 3000], and [1, 2500], respectively. The delays and targets [days, # of products] for Die/Package, Semi-finished goods, and Customer Warehouse inventories are [1, 5712], [1, 2856], and [1, 1787], respectively. The customer input demand is set to vary between 939 and 968 starting from day 61 for 577 days (see Figure 11).

The MPC model controls the inventory levels while minimizing changes in the manipulated factory starts. The optimization model uses the inventories to absorb stochas-

ticity present in the factory models while meeting customer demand. The MPC model is tuned via a suitable filter gain and forecast window for the MPC given the setting of the factory capacities. The combined tuning of the MPC and configuration of the process model handles mismatches between the discrete event model and its deterministic predictive counterpart. The MPC feedback control (enabled with a filter gain greater than zero) handles the difference between actual average and forecast demands using the system prediction model forecasting window.

We have selected the filter gains 0.01 and 0.05 for slow and fast control of the factory starts. The two sets of simulation results shown in Figure 11 use lot size equal to 20—i.e., every lot is assigned a value generated using a triangular distribution function. Fine-grain control of the factory starts is achieved with the filter gain set to 0.01. In general, while a filter gain greater than zero is necessary for having feedback, its value need to be determined judiciously in order to have an acceptable tradeoff between fast response to changes in the process model while preventing potential instability in starts of factories.

The above simulation results are consistent with those that were obtained using the Simulink/MATLAB environment. Clearly, identical behavior cannot be guaranteed between the DEVSJAVA/MATLAB and Simulink/MATLAB due to (i) the stochasticity in the factories and (ii) the discrete-event and discrete-time model specifications for the process model components. The correctness of the prototype environment, however, was verified using standard step, impulse, and sinusoidal demands where stochasticity is absent in the manufacturing process network.

7 CONCLUSIONS

Tactical (weekly, daily, or hourly) control of a semiconductor supply-chain network via model predictive control offers important and unique capabilities to decision-makers. We have presented a novel hybrid modeling approach using discrete-event and model predictive control enabled by a Knowledge Interchange Broker. The realization of this approach supports transparent and systematic specification of interactions between process dynamics and control decisions without relying on any single monolithic modeling paradigm. From a multi-formalism modeling perspective, we can employ high-level model composition as opposed to embedding model interactions inside the process and control models as is required when using interoperability in combination with model exchange standards based on XML and its current and proposed extensions (XML 2004).

A feature of the DEVS/MPC environment is its inherent support for scalable model composability and therefore simulation interoperability—i.e., not only can manufacturing process networks and model predictive control grow in their complexity, but also in their interactions (data and control message mappings and transformations). Furthermore, this approach can lend itself toward domain specific modeling which is becoming appealing for complex, large-scale domains such as the one considered in this paper.

Investigation is underway to enable the presented knowledge interchange broker to support asynchronous data and control. With this capability, we could build an environment where tactical and strategic decision control could be composed with manufacturing process networks.

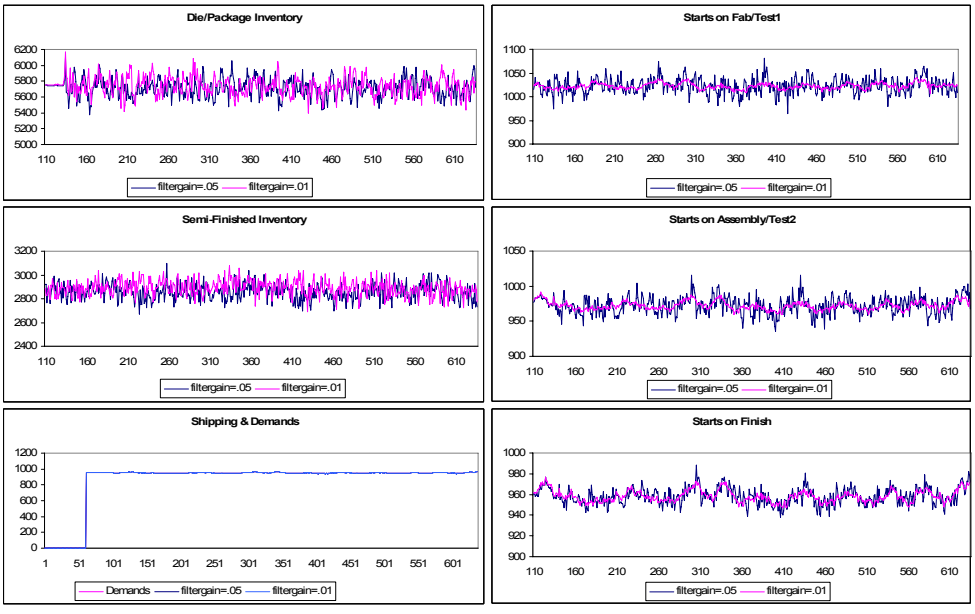

Figure 11: Simulation Plots of Inventory Levels and Factory Starts

Moreover, asynchronicity may offer a flexible basis for distributed and efficient large-scale semiconductor supply-chain system simulations. Finally, customized aggregation and disaggregation models are important for handling user-level domain-specific transformation for the semiconductor supply chain. This capability in combination with visually driven user interfaces would make this approach accessible to a larger community of stakeholders including decision-makers.

ACKNOWLEDGMENTS

This research has been supported by the NSF Grant No. DMI-0432439 and grants from Intel Research Council since 2003. Their support is gratefully acknowledged. We would like to thank the members of the Semiconductor Supply-Chain research group at ASU/Intel including Dieter Armbruster and Christian Ringhofer of the Mathematics Department at ASU and Kirk Smith at Intel Corporation.

REFERENCES

Davis, P. K. and R. H. Anderson. 2004. Improving the Composability of Department of Defense Models and Simulations. Santa Monica, CA: Rand.

DEVSJAVA. 2002. DEVSJAVA Modeling & Simulation Tool. http://www.acims.arizona.edu/SOFTWARE [Accessed February, 2005].

Godding, G. W., H. S. Sarjoughian and K. G. Kempf. 2004. Multi-formalism modeling approach for semiconductor supply/demand networks. Proceedings of the 2004 Winter Simulation Conference, ed. R. G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Kempf, K. G. 2004. Control-oriented approaches to supply chain management in semiconductor manufacturing. Proceedings of IEEE American Control Conference, Boston, MA.

Mathworks. 2002. MATLAB. http://www.mathworks.com/.

Mosterman, P. J. and H. Vangheluwe. 2002. Guest editorial: Special issue on computer automated multi-paradigm modeling. ACM Transactions on Modeling and Computer Simulation. 12(4): 249-255.

Müller, S. 2002. JMatLink. http://www.held-mueller.de/JMatLink/. [Retrieved November 2004].

Sarjoughian, H. S. and F. E. Cellier, eds. 2001. Discrete Event Modeling and Simulation Technologies: A Tapestry of Systems and AI-Based Theories and Methodologies, Springer Verlag.

Sarjoughian, H. S. and J. Plummer. 2002. Design and implementation of a bridge between RAP and DEVS. Tempe, Arizona, Computer Science and Engineering, Arizona State University: 1-26.

Singh, R., H. S. Sarjoughian and G. W. Godding. 2004. Design of Scalable Simulation Models for Semiconductor Manufacturing Processes. Summer Computer Simulation Conference, San Jose, CA.

Swaminathan, J. M., S. F. Smith and N. M. Sadeh. 1998. Modeling Supply Chain Dynamics: A Multiagent Approach. Decision Sciences, 29(3): 607-632.

Vanderbei, R. J. 1999. An interior point code for quadratic programming. Optimization Methods and Software 11: 451-484.

Wang, W., D. E. Rivera and K. G. Kempf. 2005. A novel model predictive control algorithm for supply chain management in semiconductor manufacturing. American Control Conference, Portland, OR.

XML. 2004. eXtensbile Markup Language. http://www.w3.org/XML/.

Zeigler, B. P., H. Praehofer and T. G. Kim. 2000. Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems, Academic Press.

AUTHOR BIOGRAPHIES

HESSAM S. SARJOUGHIAN is Assistant Professor of Computer Science & Engineering at ASU. His research interests are in modeling frameworks including multi-formalism and collaborative approaches, agent-based simulation, and software architecture. For more information visit http://www.acims.arizona.edu.

DONPING HUANG is a PhD student in the Computer Science and Engineering department at ASU. Her research includes modeling and simulation of distributed supply-chain networks and software design. She can be contacted at dongping.huang@asu.edu.

GARY W. GODDING is a Technologist at Intel Corporation and a PhD candidate in the Computer Science and Engineering department at ASU. His research includes modeling & simulation of semiconductor supply-chain systems, software architecture, and artificial intelligence. He can be contacted at gary.godding@intel.com.

WENLIN WANG is a PhD candidate in the Chemical and Materials Engineering Department, ASU. His research focus spans model predictive control for semiconductor supply-chain systems and control theory of complex processes. He can be contacted at wenlin.wang@asu.edu.

DANIEL E. RIVERA is Associate Professor Chemical and Materials Engineering, ASU. His areas of research include control-oriented approaches to supply chain management and scalable enterprise systems, system identification, and advanced control concepts. He can be contacted at daniel.rivera@asu.edu.

KARL G. KEMPF is Director of Decision Technologies at Intel Corporation and an Adjunct Professor at ASU. His research interests span the optimization of manufacturing and logistics planning and execution in semiconductor supply chains including various forms of supply chain simulation. He can be contacted at karl.g.kempf@intel.com.

HANS D. MITTLEMANN is Professor of Mathematics at ASU. His research focuses on nonlinear optimization, partial differential equations, and their combination. He can be contacted at mittelmann@asu.edu.