Proceedings of the 2007 Winter Simulation Conference
S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, eds.

APPLICATION OF COMBINED DISCRETE-EVENT SIMULATION AND OPTIMIZATION MODELS IN SEMICONDUCTOR ENTERPRISE MANUFACTURING SYSTEMS

Gary Godding
Hessam Sarjoughian

Arizona Center for Integrative Modeling & Simulation
Computer Science and Engineering Department
Arizona State University
Tempe, AZ 85281-8809, U.S.A.

Karl Kempf

Corporate Planning and Logistics Group
Intel Corporation
Chandler AZ, 85226, U.S.A

ABSTRACT

It is a common practice to use simulation for validating different types of control and planning algorithms. However, the science of how to rigorously integrate simulation and decision models is not well understood and becomes critically important as the complexity and scale of these models increase. In our research, we have developed a methodology for integrating different types of models using a Knowledge Interchange Broker (KIB). In this paper we describe a supply-chain semiconductor application where the KIB has been used as an integral part of developing and deploying a commercial Model Predictive Control model for use in operating a multi-billion dollar supply chain. The simulation based experiments facilitated developing and validating the controller design and data automation for a real-world semiconductor manufacturing system.

1 INTRODUCTION

The mounting complexity and scale of semiconductor manufacturing supply-chain systems have demanded advances in contemporary modeling and simulation approaches, tools, and practices. A key concept for handling complexity of discrete-part supply-chain systems is to partition them in ways to allow modeling each separately. A particular requirement of operating these types of supply-chain systems is to account for interactions among manufacturing and decision systems.

To achieve optimal inventory of parts, efficient processing of manufacturing units and delivery of products to their destinations in a cost-effective manner, a variety of process and planning models are needed (Kempf 2004). A class of semiconductor supply-chain system models have been developed using Discrete Event System Specification (DEVS)(Zeigler, Praehofer, and Kim 2000), Model Predictive Control (MPC) (Qin and Badgwell 2003), and Knowledge Interchange Broker (KIB)(Sarjoughian 2006). These models described in multiple formalisms can be composed and executed together through the KIB to reveal dynamics of the supply-chain system parts and their interactions.

The previous models of multi-formalism semiconductor supply-chain systems have been developed in laboratory settings with reduced scale and scope. They were shown to exhibit behavior consistent with the real-world systems by carrying out a suite of experiments. Although these integrated models have been crucial for demonstrating the viability of KIB for semiconductor supply-chain networks, their use in industry had not been undertaken previously.

In this paper we describe a simulation/optimization testbed developed to enable the specification and testing of a production MPC model. We show the application of the systematic integration of DEVS and MPC models using a KIB model has resulted in an environment that enabled the specification of a MPC suitable for operation in an actual semiconductor supply-chain system. The resulting MPC configuration was deployed in an industrial scale pilot study to control actual material flow over a period of several months. We conclude with an analysis of the role of KIB in a real world operational setting, the lessons learned, and future work.

2 BACKGROUND

Simulation and decision science are two distinct disciplines with many facets of ongoing research across different application domains. In the domain of semiconductor manufacturing, a variety of approaches have been devised to integrate discrete event simulation and optimization models. Among existing works, the most common approach is to implement ad-hoc interfaces to allow models written in different programming languages to exchange input and output via custom software.

Unlike those ad-hoc techniques, an approach has been developed where disparate model specifications using different execution algorithms can be rigorously composed using the Knowledge Interchange Broker (KIB). The con-

cept behind this approach is to match model semantics and enable execution interoperability through the KIB (Figure 1). This concept has been applied for the class of Discrete Event System Specification (DEVS) integrated with Reactive Action Packages (Sarjoughian and Huang 2005), Linear Optimization (Godding, Sarjoughian, and Kempf 2004), and more recently Model Predictive Control (MPC) (Huang and Sarjoughian 2006).

In the case of DEVS and MPC, the Knowledge Interchange Broker accounts for specificities of the DEVS and MPC structural specification and dynamical behaviors. The $KIB_{DEVS/MPC}$ model specification accounts for combining the DEVS and MPC models and thus ensures the correctness of their integrated structures and behaviors. The $KIB_{DEVS/MPC}$ execution algorithm accounts for the combined execution of the DEVS simulator and the MPC solver in such a way that it can correctly execute the DEVS and MPC model specifications. Separating model composability and execution interoperability is found to be key for modeling of complex interactions among models that are described in disparate modeling formalisms. The interactions account for simple and complex data transformations and synchronous execution of DEVS and MPC models.

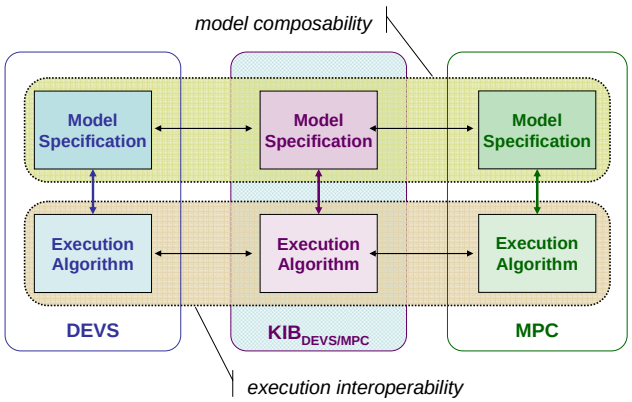

Figure 1: Integration of DEVS and MPC models with KIB

3 PROBLEM DESCRIPTION

The MPC must be tested before it can be put into production. This is to assure correct operation and avoid unplanned disruption to the business. We have developed detailed and realistic simulations of the manufacturing lines. How to integrate these DES models with production controllers is the challenge.

First, the I/O for the two models is very different. The control model requires an initial state to be populated into one set of variables and the results to be read from another whereas the DES reads and writes data via event messages. Second, the granularity of data sent between the models is generally very different. In semiconductor planning / manufacturing types of problems, manufacturing systems create data in discrete batches of lots over small time intervals (hourly asynchronous), whereas the decision models generally needs an aggregated view of the data over longer intervals (days/weeks) of time. Third, coordination of execution between the two models must be addressed. We must consider when to run the control model in respect to the simulation and vise versa. This requires the synchronization of solver runs and simulation events.

The above requirements suggests that the integration methodology must enable the mapping of different types of data structures, provide aggregation/disaggregation data transformation capabilities, and enable a flexible synchronization capability. These are the capabilities afforded by the KIB approach and exemplified below.

3.1 Example Problem

Figure 2 shows a semiconductor manufacturing topology consisting of two fabrication factories, two assembly warehouses, and two semiconductor assembly test (AT) sites being controlled by a wafer shipping decision system. The factories can ship their products to the two assembly warehouses. Material from the warehouses can be released into semiconductor assembly test. When and how much product being released from the assembly warehouses is determined by starts schedules from the associated assembly test site.

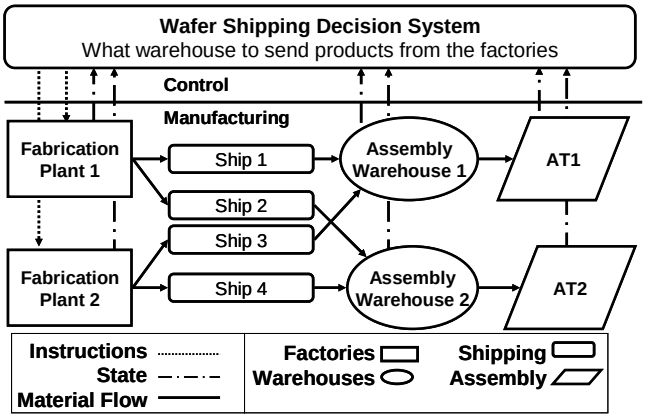

Figure 2 – Example Problem

A controller is connected to make decisions on routing material leaving the fabrication factories. The objective of the controller is to keep the warehouse inventory within upper and lower control limits. Input states to the controller are product that was shipped, warehouse inventory levels, forecasted builds of the fabrication plants, and forecasted starts from the assembly test sites. The output of the controller are commands that dictate the quantity of each product to be shipped from a given fabrication factory to the assembly test warehouses.

This example problem has several sources of stochastic behavior. The fabrication and assembly processes have

stochasticity in the throughput time and yield. This introduces error in the forecasted supply and starts schedules seen by the controller. The shipping has variability in transit times to different world geographies. This causes stochastic arrival times of material to the assembly warehouses. And finally, the boxes of lots leaving the fabrication plant are variable in size due to yield differences. Lot sizes are not adjusted prior to shipping so the quantity of material shipped to a material warehouse may be different than what was requested by the controller. For example, the controller may instruct 200 units be sent, however, if there are 3 lots of quantity 90, all 3 lots would be sent resulting in an actual ship quantity of 270. The third lot would not be split to match the actual requested quantity.

The material in the manufacturing system exhibit forward flow, however their dynamics are implicitly controlled through feedback effects via the controller (decision system). The decision system uses feedback from historical events and current state to generate instruction on what to do in the near future. Combination of feed-forward and feedback flows creates complex dynamics in the individual manufacturing and decision systems and across the supply-chain system.

The types of studies we want to carry out are to find the right data feeds, data granularity, and control frequency to effectively keep the warehouse levels within desired limits. This implies the simulation environment needs to support experimentation with data interfaces to the controller at different synchronization frequencies. Two types of questions we could answer would be the granularity of forecast data (e.g. weekly and daily) and how often do we need to rerun the control (daily, shiftly, or hourly).

Using our multi-formalism modeling approach, this problem is separated into three different models. The controller is modeled and implemented using MPC. The manufacturing topology is modeled using DEVS. The data and control integration is modeled using the KIB.

3.2 Integration Example

We will work through an integration example based on the model in Figure 2 using MPC and DEVS. The MPC will send instructions to the DEVS simulation based on state values it has received from the simulation. The integration requires modeling of instructions going to the fabrication plants and all required state messages back to the MPC. How to coordinate the execution of the MPC and DEVS simulation also needs to be modeled.

3.2.1 Manufacturing Integration Data

The fabrication plant processes material in batches or lots. Each lot has a quantity and product name. The output states for factory ships and warehouse inventory are defined in terms of collections of lots. The lot has a unique name and a quantity of one type of product. A structure for a lot is defined as:

```
Lot Structure
   Name: Unique identifier
   ProductName: String
   Quantity:  Integer
```

The data for the supply forecast is specified as a vector of material currently in the factory. The factory can be divided into a number of 'buckets'. For example, the factory could be divided into two buckets, material in the front and back halves of the factory. The controller can use this vector to predict supply.

The demand forecast is a vector giving the controller a view into the future orders. This vector is specified in time units. In the real world this vector would be supplied by the assembly test sites as future starts schedules. In the simulation it is generated from a distribution.

3.2.2 Controller Integration Data

Assume the controller outputs a matrix of quantities for each factory that specifies how much to release of each product and where it should be shipped. For the model shown in Figure 2 there would be two output matrices, one for each factory.

An example of how this matrix could look is shown below (1). The matrix is a $3 \times n$ vector that specifies a product number, a destination warehouse number, and a quantity where n is the number of product and destination combinations.

$$\begin{matrix} \text{Product} \\ \text{Destination} \\ \text{Quantity} \end{matrix} \begin{bmatrix} P_1 & \dots & P_n \\ D_1 & \dots & D_n \\ Q_1 & \dots & Q_n \end{bmatrix} \quad (1)$$

The product number is mapped to one of the products built by the factory. The destination number specifies which warehouse the material should be shipped to, and the quantity specifies how much.

3.2.3 Data Transforms

The control model requires data to be input in terms of units. For some types of data such as factory shipments, the quantity of product that leaves must be aggregated over a controller interval. For example, if the controller interval is one day, the quantities from all lots that left the factory in the previous day would be aggregated into a single value for input into the controller on the start of the current day. The value is calculated as shown in equation (2):

$$f(p,d) = \sum_{lot \in LotsOut} lot_{p,d,t\ quantity} \quad (2)$$

where p ∈ products, d ∈ destinations, previousControlTime < t ≤ currentControlTime.

If the controller interval is daily and the manufacturing model runs hourly, the controller instructions need to be disaggregated. For the mapping of the controller release to the simulation, lets assume that the value needs to be divided equally over each of the simulation time intervals. For example, if the simulation is running at an hourly granularity and the controller is generating instructions once a day, the controller instruction would be divided by 24. In general, the disaggregation could be more complex, but for illustration this simplified algorithm will be used. The equation for the equally divided disaggregation is:

$$g(p,d) = cr_{p,d,q} * \left(\frac{cf}{sf}\right) \quad (3)$$

where g(p,d) is factory release quantity for product p going to destination d, cr is controller release quantity, cf is controller frequency, and sf is simulation frequency.

3.2.4 Mapping between Vectors and Events

We must now consider how the data and control will transferred between the two formalisms. For the discrete event simulation we must read and write events to a running simulation. For the controller, we must populate input variables, initiate a solver run, and then read the output variables.

Suppose the simulation is running at hourly granularity and the controller is generating instructions once a day using the format shown in (1). Next lets assume the controller outputs a matrix of quantities specifying how much to release of each product and where it should be shipped for factory 1 as shown below:

$$\text{Factory1Ships} = \begin{bmatrix} 1 & 1 & 2 \\ 1 & 2 & 1 \\ 1000 & 1000 & 500 \end{bmatrix} \quad (4)$$

The controller output instruction needs to be mapped to simulation input events. The simulation input events have the following structure:

```
ReleaseEvent(
    SimulationPort,
    Data(product, destination, quantity))
```

The simulation port specifies where the event should be directed to. The data has three elements, what product this event is for, what destination warehouse the product should be shipped, and the quantity to ship.

To accomplish the mapping and transform we need to first consider which entity the controller output matrix is for. Since the name of the matrix is `Factory1Ships`, we can assume it is for `factory1`. This must be explicitly mapped in the KIB integration model. The controller output matrices with name Factory1Ships will be mapped to DES events going to the release input port for factory one. Next, each column of the matrix needs to be transformed into simulation input events. Each matrix element [column i, row j] is a positive scalar value (e.g., Factory1Ships [0,2]=1000). We will work through the first column in matrix (4). This column shows that 1000 units of product 1 should be sent to warehouse 1. Lets assume that both the controller and simulation use the same number scheme for products and destinations. Since the controller is running once a day and the simulator running hourly, we will need to generate 24 simulation events for each hour of the simulation on that day. The quantity will need to be transformed using equation (3). The transformed quantity value for the first column in (4) would be 1000 * 1/24. The simulation data event values in the 24 generated events would be: data(1,1,1000/24). The mapping between structures would look like:

```
ReleaseEvent(FactoryOneReleasePort,
    Data(Factory1Ships[0,0],
         Factory1Ships[1,0],
         g(Factory1Ships[0,0],
           Fatory1Ships(1,0])))
```

Pictorially, the mapping of data between the two models is shown in Figure 3. Read and writes of the data are happening between the simulation event structures and the controller matrices across different time scales. In addition, the data is being transformed to match the semantics of the target model.

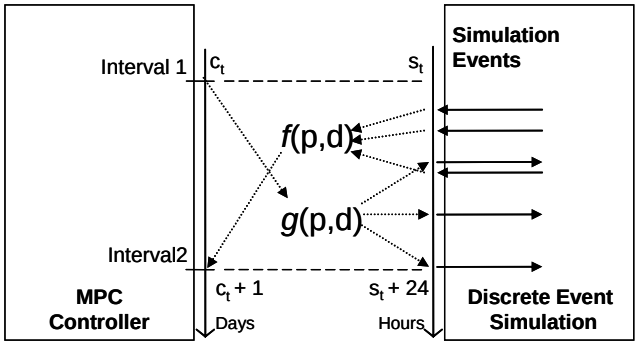

Figure 3. Transforms over Different Time Scales

3.2.5 Data and Time Synchronization

For the experiments, we require that the models can be configured to run in multiples from the other. For example, we could have the simulation run every hour and have the controller to run every simulation interval, or it could run once every 8 hours, 12 hours, etc…

When we change the synchronization frequency it will impact when to send the data between the models and the aggregation/disaggregation data transformations. For example, when running the controller daily and simulation hourly, the quantity of die in all the lots that have left the factory in the last 24 hours on each control cycle need to be added. If we were to change the frequency to control once a shift (8 hours), then we would only need to sum the die over the last shift. Conversely for the factory release commands, in the daily control cycle, 24 events would need to be sent. In the shiftly control scenario, only 8 events sent per control cycle.

4 KIB APPROACH

To integrate the models we use a Knowledge Interchange Broker (KIB). The KIB provides a methodological way to integrate multi-formalism models. The KIB enables the modeling of data transforms, mapping of data elements, and the specification of control. Figure 4 shows the conceptual view of a KIB model applied to the model shown in Figure 2.

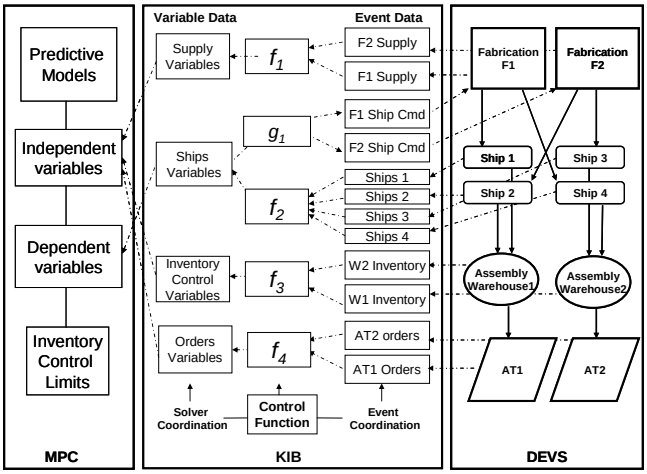

Figure 4. Composition of DEVS and MPC with KIB

The KIB allows the data integration to be modeled independently from the other models. In the case where we want to design a controller, it enables the experimentation of running a controller against an established detailed simulation. When changing controller interfaces to read different abstraction of the data, we can change the KIB model independently of the simulation model.

4.1 KIB Mapping and Transforms

The KIB must support both the mappings of fields within data structures and transformation of the data. The data in an array entry may need to be mapped to a record field, but the array data may also may need to be transformed to a different abstraction.

The general requirements for mapping between the source and target transforms in our supply network problem are:

- Multiple source data structures can be mapped to the same target structure. An example would be arrays from two different sources need to be mapped to single target set structure.
- One source structure can have multiple targets. Example is when elements of a source set need to be mapped to multiple target variables.
- Different fields of a source structure may map to different target structures
- Multiple mappings and transforms must be configurable on the same data.

In the following two sections we will describe the mappings and transformations needed to implement our DEVS/MPC simulation test environment.

4.1.1 Mappings

This section lists the data mappings that had to be provided by the KIB configuration modeling language.

UnorderedSetToArray: This mapping copies an unordered set of tuples to an ordered array. One of the data fields in the set has to be specified as an index field. The transform uses the index data values to do the ordering.

ArrayToUnorderedSet: An array of values is copied to a set of unordered tuples. The array index value can be copied to a set tuple field. If the array is part of structure, the other fields of the structure will be copied to one of the set tuple fields. For example, if you have a structure that contains field for product name and another field that contains an array of *n* values, a set can be created with *n* tuples where each tuple contains the product name, one of the array values, and the index for that array value. The starting value for the index can be specified.

ArrayValueToVariable: This mapping copies a specific array value to a single variable. The array index needs to be specified. Also, if the array is part of a structure, a key field with its matching data value can be specified. An example is would be a structure that contains a product name and an array of values. You could specify that array entry 3 of productX be written to this variable.

VariableToArrayValue: This mapping copies a variable value to a specific array entry. An index number needs

to be specified. Also, a key entry can be specified for arrays that are part of a structure.

SetFieldValueToVariable: This mapping copies a specific field value to a single variable. A key field needs be specified to determine which set tuple to use. For example, if you have a set of tuples where each contains a product name and quantity, you can specify that the quantity for productX be copied to this variable.

VariableToSetFieldValue: This transform copies a variable to specific field in a set tuple. The field name in the tuple needs to be specified, along with a key value.

Copy: The values of the fields from one model is copied into variable in the other model. The names do not need to match, just the data type. For example, integers can only be copied to integer fields.

Copy Exactly: The structure and data is copied to identical structure in other model. Names of variable and their values are maintained.

4.1.2 Transformations

This section lists the data transformations that had to be provided by the KIB modeling language.

FloatToInteger: This transforms the data value from float to integer. A rounding algorithm of floor, ceiling, or round must be specified.

IntegerToFloat: Converts an integer value to a float.

AssignValue: Assigns a value that is configured in the KIB model to a data field. This is a static value that cannot change during the execution.

Aggregations (Mean, Median, Min, Max, Sum): All these transforms aggregate multiple values into a single value. The aggregation can be for all values in the current time period, or for all values in multiple time periods. Also, if data values are in arrays, the aggregation can return an array where the entries in the returned array are aggregated from multiple arrays.

Disaggregation: Different types of disaggregation can be supported. A general purpose disaggregation is to divide the source value into equal target values. The design of the KIB enables extensions for customized disaggregation algorithms.

Scale: Multiplication or division operations can be specified to scale the data values.

4.1.3 KIB Control Modeling

The KIB is required to support a synchronization model that enables the controller to run in multiples of the simulation. Experiments were designed with daily, shiftly, and hourly control against a simulation that could run at hourly granularity. The KIB configuration model allowed the execution of either model to run in multiples of the other.

To provide this capability, the KIB had to coordinate the timing of simulation input/output events with the solver execution. It also needed to execute aggregation/disaggregation transforms across the correct time intervals. The ability to adjust the aggregation/disaggregation of data based on execution frequency is a key enabler of experimentation at different control frequencies. For example, if you want to try daily control against hourly data, you must aggregate all events that occurred over last 24 hours of logical simulation time.. If you then want to try shiftly (8 hour shifts) control, then you must only aggregate over the last 8 hours for logical simulation time.

5 PILOT EXPERIMENT

The real world supply network topology is shown in Figure 5. This topology is an extension to the model shown in Figure 2. For simplicity, shipping components are depicted as arrows. This topology has 3 factories, 27 shipping lanes, 9 warehouses, and 9 assembly sites. Each of the fabrication plants can produce up to 15 different products.

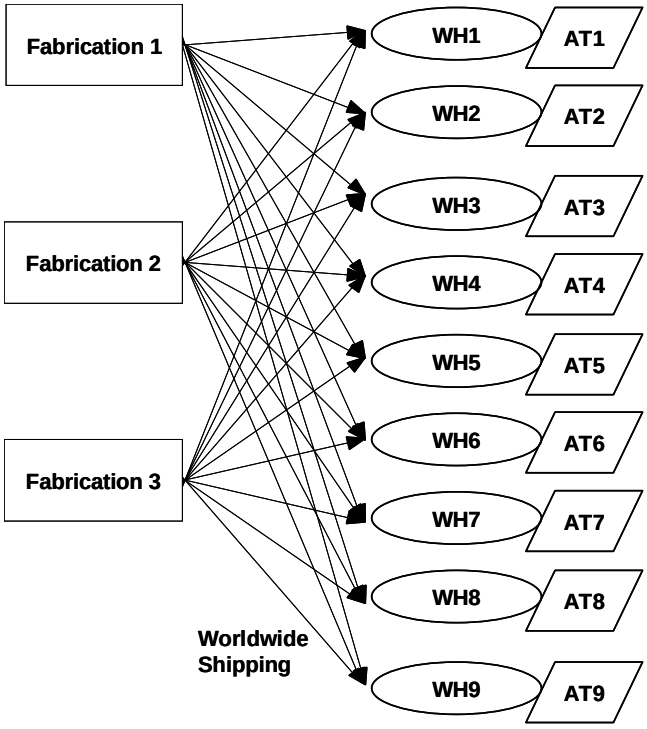

Figure 5. Real world topology

The MPC was implemented using the Honeywell Profit Suite™ set of applications. The DES had been developed using the DEVSJAVA simulation environment.

The MPC controller design required a different model instance for each product built from the fabrication plants. This resulted in 15 different product controllers that needed to run concurrently. The product controllers were coordinated using a dynamic, real-time optimizer. This optimizer

provided both dynamic coordination and steady-state optimization to the underlying 15 control applications.

A separate simulation model was connected to each of the 15 product controllers resulting in a distributed simulation. That is, each of the fifteen controllers had a designated simulation model. Each simulation matched the topology shown in Figure 5 but had different stochastic distributions configured to match each of the products characteristics for the supply and demand forecast vectors.

The KIB had to coordinate the execution of the 15 different simulations, 15 different controllers, and one dynamic, real-time optimizer. It also needed to map and transform the data between each of the simulation and controller models.

5.1 Controller Development Approach

The goal of the simulation environment was to enable the development and validation of the controller prior to putting it in production. Although this kind of simulation-based design is common practice, the use of the KIB enabled experiments and engineering of the complex interactions between discrete manufacturing processes and controller. A two step iterative process was devised (Figure 6).

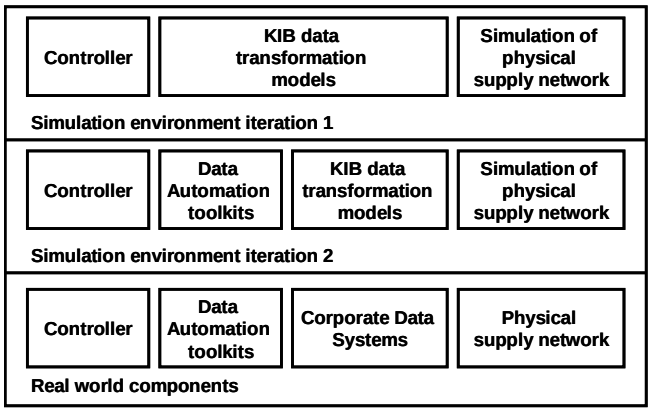

Figure 6. Simulation Iterations vs. Real World

First, the KIB model generated data that was directly read and written into the controllers variables. Second, the KIB model was revised to generate data the same way as the enterprise data systems. The production data automation toolkits would be developed against simulated data feeds.

The real world components section of Figure 6 shows the physical systems we needed to work with. There is the physical supply network, the corporate data systems that capture current states and information about the physical systems, some data automation software to transform the data into the format required by the controller, and the actual controller system.

The first iteration of experiments supported the development of the controller. Realistic stochastic simulations were run and validated against historical data. The controller was then developed and integrated with these simulations using the KIB. This stage of simulation supported the development of the controller and its required data interfaces. Base issues were worked out such as scalability and controller design.

In iteration two the KIB was changed to output data to the controller in a format that matches the corporate data systems. The production data automation toolkits were developed during this iteration. The same simulation of the physical models were used, however, the KIB data output to the MPC was different. This enabled testing and development of the production data automation toolkits for the controller.

After the two iterations of development, the controller and associated data automation toolkits were put into the production configuration.

5.1.1 Findings

On the first simulation/controller runs, it was found there would be scalability issues with the controller design and the simulation. Although each of the models ran OK in standalone mode, the integration highlighted invalid assumptions each had made about the other system. The controller had to be changed into a hierarchical design where separate instances controlled each product. Performance tuning had to be performed on the simulation to manage the large numbers of active simulation entities. Changes also had to be made in the simulation to correctly model how discrete lots are shipped from the factory. It was found this was an important behavior to simulate for the controller. The discrete nature of lot sizing errors had impacts on how the controller needed to be tuned.

In the second iteration, we changed KIB models to exactly reproduce how data is sent and received from the production data systems. This resulted in development of the data automation toolkits prior to plugging the controller into production. The KIB enabled experimentation and refinement of the aggregation needed for forecast vectors. It also highlighted some invalid assumptions on how data is provided from internal company systems versus subcontractors. The KIB provided a quick and efficient way to do experimentation with many different types of aggregation strategies.

5.2 Results

The MPC models and production data automation toolkits worked as designed on the first run in production. This was a significant accomplishment since it was the first time MPC was used in an actual production instance of a discrete semiconductor manufacturing problem.

The controller design worked with a daily update to the shipping signals. The supply and forecast vectors needed daily granularity for the first few days, and then could use weekly buckets for the next few weeks. When the controller was plugged into production, it was able to keep inventory within limits automatically at all warehouses as good as the current manual processes.

The team that developed the controller and simulation comprised of 3 engineers, two senior control engineers one software / simulation engineer. Projects of this scale typically take much more resources. Without the KIB modeling approach, the ability to experiment with different control frequencies and data sources would have been limited or impossible within the time constraints. Either more time would have been required for the simulation or less robust controller put in at start of production.

6 CONCLUSIONS

An important benefit was the combined flexibility and rapid controller design prototyping enabled with the KIB. The pilot experiment was carried out concurrent with the actual control of the production line and thus demonstrated the impact of the KIB in industrial strength setting. This approach to simulation-based design is indispensable in employing new mixed tactical and strategic operation of multi-billion dollar industries. The KIB enabled much more experimentation (e.g., validating controller) than would be normally possible in short time frames.

The KIB also allows for better experiments since it highlights the integration mismatch between the models. That is, it explicitly provides visibility to the integration issues and provides a capability to understand the issues and methodologically design solutions around them.

7 FUTURE WORK

We plan to extend the existing models onto other product lines and other segments in the supply network. This adds complications such as the increased number of planning product configurations and consideration of complex bills of material. We also plan to develop more advance aggregation and disaggregation transforms into the KIB. The advanced functions would be based on more detailed knowledge of the segments of the supply network domain being modeled.

ACKNOWLEDGMENTS

We would like to acknowledge Kirk Smith of Intel Corporation and Duane Morningred of Honeywell Corporation for their work on the MPC.

REFERENCES

Godding, G., H. Sarjoughian, and K. Kempf. 2004. Multi-formalism modeling approach for semiconductor supply/demand networks. In *Proceedings of Winter Simulation Conference*, 232-239. Washington DC, USA.

Huang, D., H. Sarjoughian, D. Rivera, G. Godding, K. Kempf. 2006. Experiment Analysis of Hybrid Discrete Event Simulation with Model Predictive Control for Semiconductor Supply Chain Systems, In *Proceedings of Winter Simulation Conference*, 1863-1870. Monterey, CA, USA.

K. Kempf. 2004. Control-Oriented Approaches to Supply Chain Management in Semiconductor Manufacturing. *Proceeding of the American Control Conference*, 4563-4576. Boston, MA, USA.

Qin, S., and T. Badgwell. 2003. A survey of industrial model predictive control technology. *Control Engineering Practice* 11 (7): 733-764.

Sarjoughian, H. 2006. "Model Composability", In *Proceedings of Winter Simulation Conference*, 149-158. Monterey, CA, USA.

Sarjoughian, H., and D. Huang. 2005. A multi-formalism modeling composition framework: Agent and discrete-event models. In *Proceedings of the 9th IEEE International Symposium on Distributed Simulation and Real Time Applications*, 249-256. Montreal, Canada.

Zeigler, B., H. Praehofer, and T. Kim. 2000. *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems.* 2nd ed. Academic Press.

AUTHOR BIOGRAPHIES

GARY W. GODDING is a Technologist at Intel Corporation and a PhD candidate in the Computer Science and Engineering department at ASU. He can be contacted at <gary.godding@intel.com>.

HESSAM S. SARJOUGHIAN is an assistant professor of Computer Science and Engineering at ASU. He can be contacted at <sarjoughian@asu.edu>.

KARL G. KEMPF is Director of Decision Technologies at Intel Corporation and Adjunct Professor at ASU. He can be contacted at <karl.g.kempf@intel.com>.