Model-driven Performance Analysis of Reconfigurable Conveyor Systems used in Material Handling Applications

Kyoungho An, Adam Trewyn, Aniruddha Gokhale
*Department of Electrical Engineering and Computer Science
Vanderbilt University
Nashville, TN 37235, USA
Email: {kyoungho.an, adam.c.trewyn, a.gokhale}@vanderbilt.edu

Shivakumar Sastry
†Department of Electrical and Computer Engineering
The University of Akron
Akron, OH 44325, USA
Email: {ssastry}@uakron.edu

***Abstract*—Reconfigurable conveyors are increasingly being adopted in multiple industrial sectors for their immense flexibility in adapting to new products and product lines. Before modifying the layout of the conveyor system for the new product line, however, engineers and layout planners must be able to answer many questions about the system, such as maximum sustainable rate of flow of goods, prioritization among goods, and tolerances to failures. Any analysis capability that provides answers to these questions must account for both the physical and cyber artifacts of the reconfigurable system all at once. Moreover, the same system should enable the stakeholders to seamlessly change the layouts and be able to analyze the pros and cons of the layouts. This paper addresses these challenges by presenting a model-driven analysis tool that provides three important capabilities. First, a domain-specific modeling language provides the stakeholders with intuitive artifacts to model conveyor layouts. Second, an analysis engine embedded within the model-driven tool provides an accurate simulation of the modeled conveyor system accounting for both the physical and cyber issues. Third, generative capabilities within the tool help to automate the analysis process. The merits of our model-driven analysis tool are evaluated in the context of an example conveyor topology.**

Keywords: **Reconfigurable conveyors, design-time analysis, model-based, simulations.**

I. Introduction

Many industrial sectors, such as manufacturing, automotive, and material handling, are increasingly moving towards adopting reconfigurable conveyor systems in their processes since they offer significant flexibility in readily adapting to newer products and product lines, while making efficient use of available space, and all of these at a fraction of cost that otherwise would be incurred if an entirely new conveyor system is to be installed. To quote from a recent article [1]: *the key factor in a truly reconfigurable modular conveyor system is the ability to connect and reconnect a wide variety of modules and accessory modules that allow engineers the freedom to tweak production lines when necessary without the cost of a brand new conveyor or the risk of losing the conveyor's integrity.*

When faced with the task of using reconfigurable systems in businesses, such as a material handling system used in facilities like FedEx, UPS, and baggage handling in airport terminals, engineers and layout planners for the conveyor systems must often grapple with numerous questions including but not limited to: What is the maximum sustainable rate of flow of goods in the system? Can handling of certain types of goods be prioritized over others? Does a certain layout of the conveyor system lead to starvation of certain paths in the system? What is the impact of failures of certain sections of the conveyor system on the overall throughput and hence the monetary costs? How to plan the inter material spacing on the conveyors such that the goods do not collide when they are switched through transfer elements (called turnarounds)?

A naive solution based on trial-and-error does not scale when dealing with large deployments. A straightforward application of techniques such as combinatorial optimization or queuing theory in isolation do not suffice either for the following reason. The intertwined relationships between the cyber elements, *i.e.,* the micro-controllers that regulate each unit and the wireless transceivers that provide communication links between micro-controllers in physically adjacent units, and the physical transfer of parts over the conveyor units present formidable challenges in readily finding answers to the above questions.

Answering the questions faced by the engineers and layout planners obviously requires a design-time solution in contrast to the need for physically deploying a system and iterating over multiple possibilities. A critical requirement for such a design-time "what-if" analysis capability is the need for it to account for in tandem both physical artifacts of a conveyor system (*e.g.,* speed of belts, inter-material spacing, size and type of the material being handled, response time of commands to control belt motor speeds, rate of flow of material into the input source of the system, and sensors that scan moving goods) and cyber artifacts (*i.e.,* message formats and signaling protocols between the individual units of the reconfigurable system, timing of the messages, and synchronization policies among the highly concurrent executing software artifacts).

Model-driven performance analysis [2] of the reconfigurable conveyor cyber physical system (CPS) [3] provides a promising solution [4] to address these requirements. In particular,

our model-driven analysis tool comprises three primary artifacts:

1) A domain-specific modeling language [5] within our tool provides intuitive abstractions to engineers and layout planners to describe the proposed layouts of their system without unduly tightly coupling their intentions to any specific analysis capability.
2) An analysis engine we designed that implements the behavior of the conveyor units in the MATLAB Simulink/Stateflow simulation engine by tightly integrating the cyber and physical aspects of the conveyor system along with the critical timing properties [6], [7].
3) A generative capability [8] that synthesizes artifacts for the analysis engine and helps to completely automate the design-time analysis process.

By augmenting the simulation with synthetic data that is derived from our experience with real conveyor systems, we validate the design of the conveyor system layout using our model-driven analysis tool. This approach allows us to evaluate several system-level performance parameters, such as throughput and end-to-end latency at design-time thereby providing insights into the efficiency of the proposed layout to meet the business objectives.

The remainder of this paper is organized as follows. Section II compares our work to related research; Section III describes the system model of our reconfigurable conveyor system alluding to the kinds of material that we consider flowing on the conveyors; Section IV presents the design of the model-driven analysis framework for the reconfigurable conveyor system; Section V presents results evaluating our tool on an example topology; and finally Section VI offers concluding remarks and next steps.

II. Related Work

Although a large literature in model-driven engineering of large-scale systems exists, in this section we present related research in the field of reconfigurable systems focusing primarily on those works that deal with assessing performance of the system along different metrics. Moreover, we present works that are closely related to the different aspects of our work. We also noticed that most related research discusses reconfigurable manufacturing systems, which is a more general class of systems that encompass reconfigurable conveyor systems.

A related work closest in spirit to ours appears in [9]. The authors describe a discrete event perspective of reconfigurable manufacturing systems. As in our case, they too use model-driven engineering [10] principles to describe the layout of the system. Moreover, they also distinguish between the cyber and physical aspects of the system. This related research, however, focuses on developing mathematical models to conduct criticality analysis of different configurations (*i.e.*, layouts) to determine the best configuration for a given set of product mix while also satisfying other constraints, such as cost. Overall, this research has similar goals to ours; the analysis approach used and metrics evaluated are different.

Another work closely related to ours appears in [11]. The motivation of the work is to assess the reconfiguration in layout of a manufacturing systems as the product mix changes. The metrics used to evaluate the layout include material handling costs and operational performance factors. The authors of this work use open queuing networks to develop a mathematical model of the system. A related recent work that uses Petri Nets to develop analytical models of reconfigurable systems appears in [12]. Despite the similarity in the goals of these related works, to the best of our understanding, these works do not explicitly account for the tight integration between the cyber and physical issues. Instead, performance estimates are collected based on expected arrival patterns of parts.

In [13], the authors present a framework to assess reconfigurability of manufacturing systems. Although this work applies to a more broader range of reconfigurability than our work, it is more focused on assessing the manufacturing degrees of freedom, *i.e.*, identifying the different ways in which a product can be manufactured. Thus, although both approaches pertain to design-time analysis and are model-based, the goals and outcomes are quite diverse.

A tutorial on reconfigurable modular systems focusing primarily on pallet-based conveyor systems is presented in [14]. The relevance of this related work to ours stems from the fact that the authors present a broad range of metrics to assess flexibility of the system. Among the list provided by the authors, our work focuses on evaluating the performance of layout modifications and assess scalability.

Verification of the logical controllers in reconfigurable systems is considered in [15]. The authors use the concept of timed transition models to model the behavior of the controllers to verify its properties. The authors also use their technique to iteratively arrive at a desirable controller for the reconfigurable system. Our work is orthogonal to the goals of this related work in that we are concerned with measuring different performance factors of a given layout and controllers while the related work focuses on the synthesis and verifying the correctness of logical controllers for reconfigurable conveyor systems.

Li et. al [16] describe an approach to model reconfigurable manufacturing system focusing primarily on how to update the models to capture reconfiguration decisions. In particular, they use Petri Nets to describe the behavior and introduce the notion of *net rewriting* that is used within a model transformation process to update the models in accordance with the changes in the reconfiguration. While not directly related to the evaluation goals of our work, this work is related to model-driven engineering and generative aspects of our work. It is conceivable that our future work may benefit from these transformations.

A software engineering perspective of reconfigurable manufacturing system is presented in [17]. The authors use the Microsoft COM model to build a software component-based design of a reconfigurable manufacturing systems. The component-based approach makes it easier to achieve the plug-and-play vision of reconfigurable systems.

Despite several existing efforts in evaluating different performance factors of reconfigurable manufacturing systems, we believe that related research does not provide a holistic cyber-physical systems perspective of reconfigurable manufacturing systems. In our research we develop an analysis engine that tightly couples the cyber and physical parts to provide accurate performance data in a design-time tool.

Our prior work in the area of reconfigurable conveyor systems has focused on analyzing the reliability of the software controllers [18], providing efficient mechanisms for monitoring and diagnostics [19], and preliminary work on evaluating worst case end-to-end response time in composable conveyor systems [20]. The research presented in this paper expands on our earlier work focusing on the design and implementation of an automated, design-time analysis tool for performance analysis of reconfigurable conveyor systems.

III. Model of Reconfigurable Conveyors

The reconfigurable conveyor systems we consider in this paper move parts from one or more inputs, $\mathcal{I}$, to the outputs, $\mathcal{O}$. These systems are composed using two kinds of units — *Segments* and *Turnarounds* which are illustrated in Figure 1 — that have fixed behaviors [21]. Each unit is autonomously regulated by a local microcontroller that interacts with microcontrollers in physically adjacent units over wireless links to coordinate the transfer of parts from one unit to another.

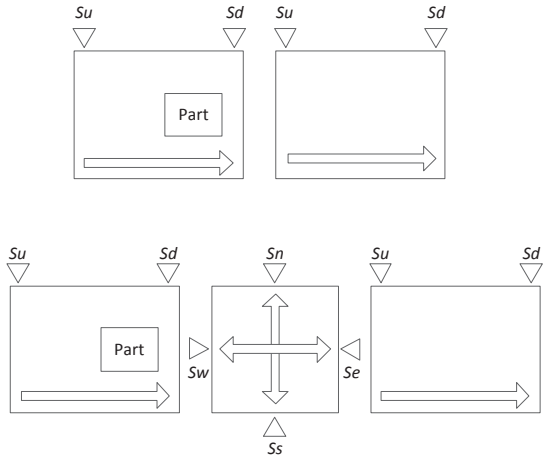

Fig. 1. Segment and Turnaround

A Segment moves a part over a fixed distance, in one of two assigned directions. Input and Output units are halves of Segment units that can move parts in one direction only. S_u and S_d are, respectively, upstream and downstream sensors at each segment that are activated as a part moves into its scanning range.

A Turnaround unit has four ports; each port can be configured either as an input port or as an output port. For each port there exists a sensor. In the figure, the suffixes denote the direction (east, west, north, south). To keep the presentation simple, we assume that a Turnaround can handle only one part at a time while a Segment may contain multiple parts spaced some distance apart as they flow from one end to the other. When two or more parts simultaneously arrive at different input ports of a Turnaround, it can accept only one of the parts.

A specific composition of instances of the above kinds of units is a conveyor system. Figure 2 shows an example of conveyor systems obtained by composition.

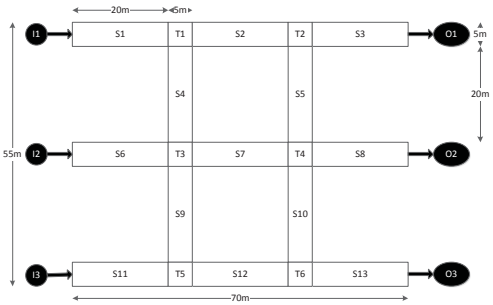

Fig. 2. Reconfigurable Conveyor System

We view the conveyor systems, intuitively, as graph $G = (U, E)$. The nodes of G, *i.e.*, $u_i \in U$, represent the units — Segments, Turnarounds, Inputs, and Outputs. An edge $(u_i, u_j) \in E$ represents the relation that a part can be transferred from u_i to u_j. Parts that arrive via input $I_k \in \mathcal{I}$ are delivered to a specific output $O_j \in \mathcal{O}$ along a path $P(I_k, O_j) =< u_1 = I_k, u_2, \cdots, u_n = O_j >$, $u_i \in U$. These paths can be pre-determined via offline analysis. Specifically, routing functions to determine the paths are generated and deployed at each Turnaround at design-time.

We view the path, $P(I_k, O_j)$, along which a part moves from an input $I_k \in \mathcal{I}$ to an output $O_j \in \mathcal{O}$ as a *channel*. Parts arrive sporadically at I_k with a minimum inter-arrival time of T_k and a relative deadline D_k before which the conveyor system must deliver the part to O_j. We use τ_k^j to refer the j^{th} part that arrived via input I_k. Because multiple channels share common units, congestion can occur at these units.

To improve the throughput of the system, it is desirable to increase the processing rates and change accepting priorities of congested Turnarounds where parts are injected to the channels. Because of confluence of multiple channels, the parts are likely to experience congestion - and thereby reduction of the throughput achieved in the system. Thus, the problems of setting appropriate accepting priorities and processing rates in Turnarounds are interesting and involve competing objectives.

For the purposes of this paper, we focus on reconfigurable conveyors employed in material handling facilities, such as those found in FedEx and UPS sorting facilities as well as baggage handling facilities in airport terminals. Thus, we classify a good (*i.e.*, a part) as belonging to a small (*e.g.*, envelopes), medium (*e.g.*, small boxes) or large category (*e.g.*, large boxes).

IV. Design and Implementation of Analysis Framework

In this section we present the details of our design-time performance analysis framework. The design architecture of the analysis framework for conveyor systems is formed of structural models and behavioral models that account for both the cyber and physical parts of the system. Figure 3 shows the overall architecture of the analysis framework we have developed.

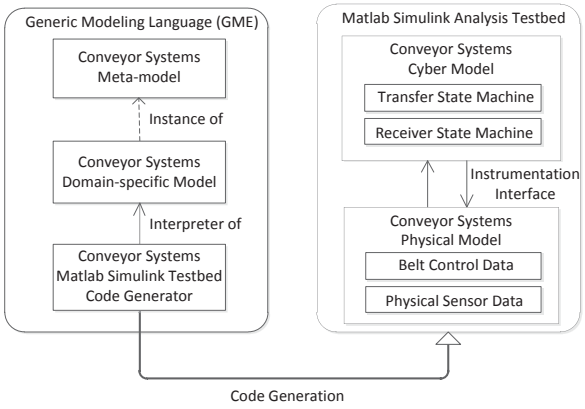

Fig. 3. Overall Architecture of Analysis Testbed

The structural models are realized at two levels. At one level, the layout of the conveyor system is modeled using intuitive domain-specific modeling artifacts provided by a modeling language we developed. At the second level, structural models of the conveyor system are represented within a simulation framework. For our work, we rely on the Matlab/Simulink suite. The layout of the conveyor system is automatically transformed into Matlab/Simulink structural models using the generative capabilities of our model-driven framework. The behavior models comprising both the cyber and the physical aspects of the conveyor system are implemented in Matlab Simulink and Stateflow.

The remainder of this section provides details on the design and implementation of our analysis framework.

A. Domain-specific Modeling and Generative Capabilities

We have used the Generic Modeling Environment (GME) [22] to develop the domain-specific modeling language (DSML) and generative capabilities for the domain of reconfigurable conveyor systems. Figure 4 illustrates the metamodel, which is at the heart of the DSML for reconfigurable conveyor systems. The metamodel of the system comprises primarily of the building blocks found commonly in a conveyor system, such as input bins, output bins, and blocks which are classified into Segment and Turnaround that help move the material along the conveyor system. The metamodel also contains connection components used to link the building blocks.

Each building block has attributes to configure the parameters used in subsequent analysis, such as length of belts, speed of belts, quantities of packages, and address of nodes. Specifically, the address attribute is used to identify the building

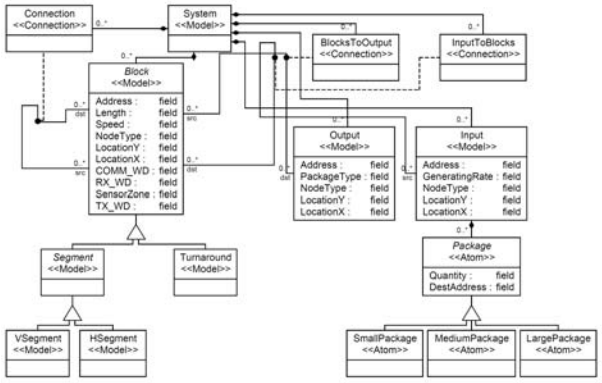

Fig. 4. Meta-model of Reconfigurable Conveyor Systems

blocks. Addresses are needed to generate routing tables at each Turnaround so that routes get set up for packages to flow through the system.

The Length and Speed attributes are used for the physical model implemented within the analysis engine. According to the physical attributes, throughput results of the systems and bottleneck points would be different in the system and bottleneck points can be found and fixed through compraring results of simulation analysis. `NodeType` in `Block` is used to differentiate the type of block which can be a Segment or Turnaround. `LocationX` and `LocationY` in Block are manipulated for proper graphical layouts. `Sensor Zone` indicates coverage of sensors in block. `COMM_WD`, `RX_WD`, `TX_WD` denotes watchdog timers used in logical controllers. The attributes explained above should be configured in each components in the domain-specific model.

When an example model using the DSML is created, the attributes defined by the metamodel are configued for performance analysis of the system. The domain-specific models of the entire layout provide analysts with a higher level of abstraction of the system that is easier to comprehend. The generative capabilities within the DSML transform these models into appropriate structural and behavioral models of the whole system in the format recognized by the underlying analysis engine, such as a simulator. In our case we rely on Matlab Simulink/Stateflow for the structural and behavioral models of the cyber physical system.

The goal of the GME-based DSML is to allow analysts to place necessary components such as input bins, output bins, Segments, and Turnarounds at desired locations and configure attributes for the building blocks. In turn the GME-based DSML transforms the GME-based model into the underlying artifact. This decoupling helps the analyst to try many different layouts and the entire process of transforming into the underlying representations is completely automated.

An example model of a conveyor system using the DSML is shown in Figure 5. There are 13 Segments, 6 Turnarounds, 3 Input bins, and 3 Output bins. The building blocks are connected to other blocks to help move material through the conveyor system. Our model specifies attributes such that each

input bins randomly generate parts (or units) catagorized into small, medium, and large categories. Input bins are annotated as $I1$, $I2$ and $I3$. For instance, entities come into the system via $I1$, $I2$, or $I3$. Entities leave the system via Output bins annotated as $Small$, $Medium$, and $Large$.

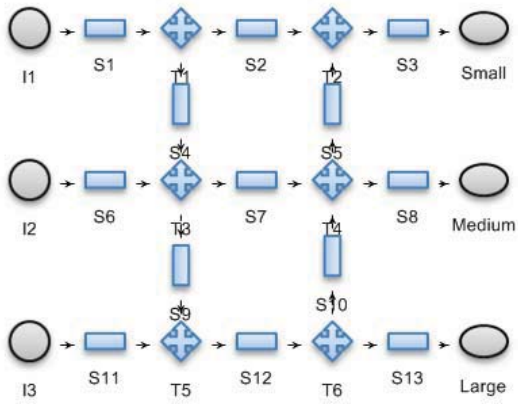

Fig. 5. Domain-specific Model of Reconfigurable Conveyor Systems

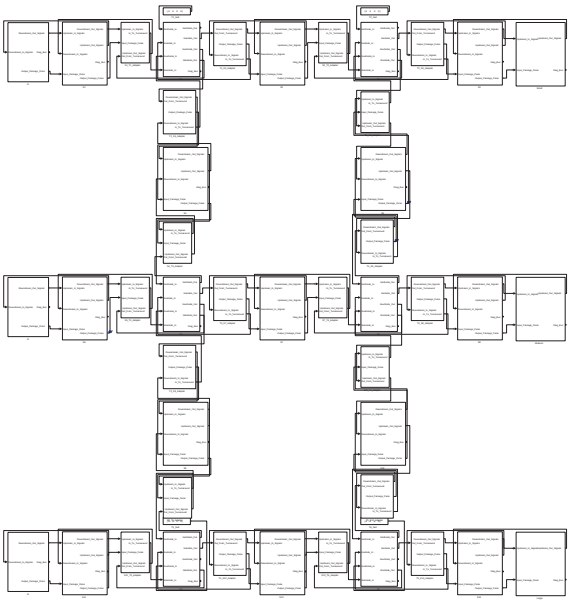

Fig. 6. Transformed Model into Matlab-based Representation

The generative capability of the DSML executes a shortest path routing algorithm and produces routing tables at each Turnaround. A small package that arrived at input bin $I1$ would move along the path: $I1$, $S1$, $T1$, $S2$, $T2$, $T3$, $Small$. A medium package would move along the path: $I1$, $S1$, $T1$, $S4$, $T3$, $S7$, $T4$, $S8$, $Medium$. A large package would move along the path: $I1$, $S1$, $T1$, $S4$, $T3$, $S9$, $T5$, $S12$, $T6$, $S13$, $Large$. Entities coming from $I1$ are deflected via $T1$, and $T3$, and $T5$. There are other possibilities for moving entities from other input bins that are not discussed in the paper.

The example layout model is then transformed into Matlab Simulink codes by the GME interpreter associated with the DSML to analyze performance of the system. Figure 6 shows the Matlab Simulink anaysis model for the example model converted by the GME interpreter. The overall layout is basically similar to the example model because it uses location attributes in the domain-specific model. Every block contains a cyber model implemented by Matlab Stateflow and the physical model implemented in Simulink. Details of the Matlab-based engine are discussed next.

B. Cyber and Physical Models in the Analysis Engine

Conveyor systems are composed of cyber parts, such as the controllers; physical parts, such as the Segments; and the interfaces connecting the cyber and the physical world. Therefore, our design of the structural and behavioral models of the conveyor system within the analysis engine needs a clear separation of cyber model and physical model to simulate conveyor systems.

Thus, the cyber controller logic was implemented as a state machine within the Stateflow toolset; this choice of environment allowed a direct translation of the controller logic from the existing state-chart model to an executable software implementation. For purposes of testing and validating the prototype controllers, a system simulator of the physical environment was also implemented within Simulink in order to simulate the physical behavior of a conveyor under the control of block controllers. Though a real conveyor system will eventually be needed to demonstrate the capabilities of a controller logic, a software simulation is being relied upon in this paper in order to allow for maximum flexibility in unit testing, compositional testing, and architectural modifications.

We developed a modularized Simulink unit called a *Conveyor Skid*, which is implemented as a self-contained unit representing both the physical and cyber components of a single segment for purposes of modeling a single conveyor block. Within a conveyor block, the Conveyor Skid exists as a single self-contained controller system as well as a single self-contained simulator block. As both the controller and simulator are intended to be self-contained, connections between these units are restricted to those intended to be present within the actual system; at this stage, these connections represent the motor control signal from controller to simulator and the sensor data feeds from simulator to controller. As such, the simulator block may eventually be removed and replaced with signal interfaces to a physical model without disrupting the implementation of the controller.

All remaining elements within the Conveyor Skid block consist solely of inputs/outputs to and from the outside of the block, as well as single-step signal delays necessary to break algebraic loops within Simulink. These input and output connections exist in order to provide for compositional simulation; that is, Conveyor Skid blocks may be connected together to form a conveyor network which may be simulated as a whole. The external connections represent external neighbor network connections between conveyor controllers, a diagnostic signal bus from the simulator, and a signal line for simulating

package handoffs between simulator blocks.

Next we describe the details of the cyber and physical models in the Conveyor Skid.

1) Cyber Model: The cyber model was implemented as a finite state machine using the Stateflow toolset in Simulink. Each Segment and Turnaround has a receiver controller logic and a transfer controller logic. Figure 7 shows a receiver state machine embedded in a controller of a Segment. Initially, a Segment waits until receiving a request message for transferring a part from a upstream block. When the Segment receives the request message from the upstream block, it checks if the upstream sensor is inactive which is for ensuring a space is available for a transferred part, and the downstream sensor is inactive which is for avoiding a conflict between the receiver machine and the transfer machine in the same Segment.

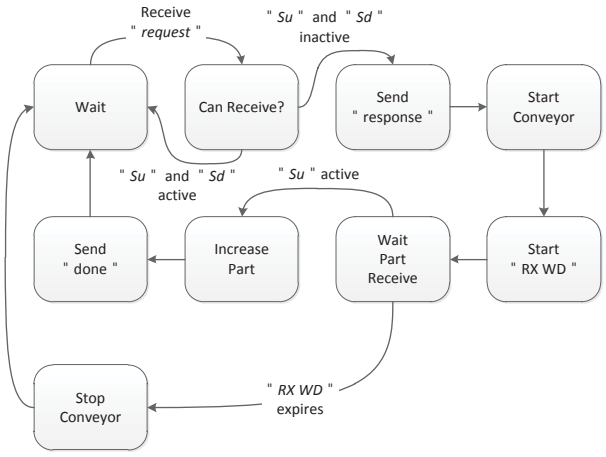

Fig. 7. Segment Receiver State Machine

If these conditions are all clear, it sends a response message back to the downstream block to notify that a part can be accpeted and it actuates the conveyor belt at the configured speed. Then, it starts a watchdog timer to wait for the transferred part. If the watchdog timer is expired, that means the part has not arrived at the Segment and a problem occurred in the middle of the process of transferring the part between the blocks. If the upstream sensor is active, it indicates a part has safely arrived at the Segment. If a part has securely arrived at the Segment, it increases the number of packages on the Segment belt and sends a finishing message to inform transferring is done well.

A transfer state machine in a Segment, which coordinates with a receiver state machine in a next block, is depicted in Figure 8. At first, the transfer state machine also waits until the downstream sensor is active and parts are on the belt. If both conditions are true, the Segment is ready for transferring a part to the next block. Therefore, it needs to be stopped until receiving a permission from the next one, so it sends a request message to obtain the permission. Here, the receiver machine may not give the permission to the given request. In that case, Comm WD, which is used for waiting for the approval from receiver side, expires and the block sends a request again to the receiver machine. If the machine receives the response message, it energizes the conveyor belt of the Segment and waits for the part to successfully transfer to the next block. If the part is normally moved out, the transfer machine can get a successful Done message from the receiver. If the message is acquired, the number of parts is decreased and the machine goes back to the initial state.

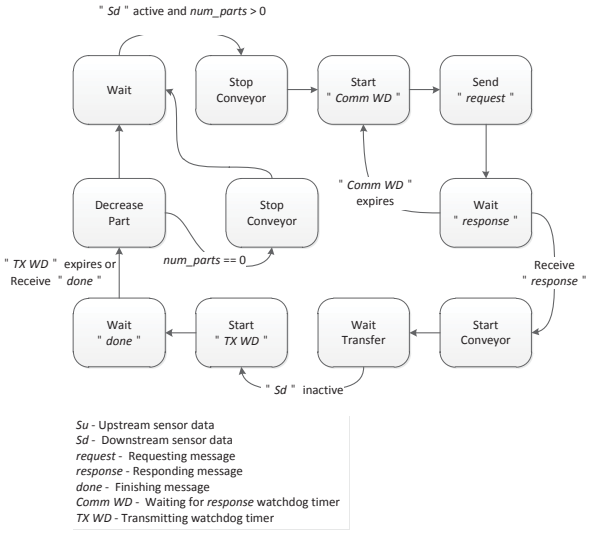

Fig. 8. Segment Transfer State Machine

Controllers within Turnarounds also incorporate a receiver state machine and a tranfer state machine as Segments do. Figure 9 represents a receiver state machine in a Turnaround. The processes of a receiver state machine of a Turnaround are similar to a receiver state machine in a Segment. A distinguishable difference of controllers between a Segment and a Turnaround is that a Turnaround has more incoming and outgoing ports than a Segment. Hence, state machines in a Turnaround should have logic to differentiate signals from varying ports.

According to the physical figure of Turnaround of Figure 1, a Turnaround owns 4 ports and accordingly holds four sensors for each direction. Sr in the receiver state machine stores a selected signal among a set of sensors called $S = \{Sw, Ss, Se, Sn\}$ from an appropriate direction. Moreover, a Turnaround takes in functions to decide a direction for each package by the *Decide Route* state using a routing function determined and deployed by the GME interpreter. Accordingly it energizes a selected actuator among $A = \{Awe, Asn\}$ and the direction of the actuator. After a route for a package is determined, a Turnaround should move the package to the center of the belt to avoid a physical collision which can occur by actuating the chosen belt before the package is totally moved in the Turnaround.

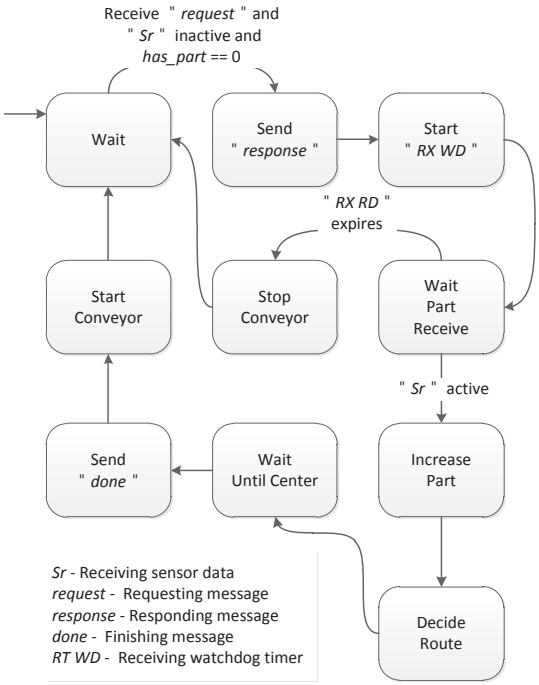

Fig. 9. Turnaround Receiver State Machine

Lastly, a Turnaround accepts only one part to simplify its logic. After processing one part, it accepts another part to be processed. The rest of the logic in a receiver state machine in a Turnaround is basically the same as a Segment. A transfer state machine in a Turnaround (shown in Figure 10) is similar to a transfer state machine in a Segment excluding that it uses St as a sensor signal which is selected among $S = \{Sw, Ss, Se, Sn\}$. Furthermore, it accepts only one part on a belt in the same way as the receiver machine. All receiver machines and transfer machines introduced above can seamlessly communicate with transfer machines and receiver machines of adjoining blocks (that are modeled in the layout model).

2) Physical Model: Next we describe how we architected the physics of the different blocks of the conveyor system. Due to the complex nature of the simulation, we have abstracted away the pertinent details in our explanation while leaving out unnecessary low-level details of Simulink building blocks we used.

Figure 11 depicts a high level perspective of the physics of simulated segment. It comprises the following building blocks:

1) *Belt Statistics Calculator:* Calculates belt odometer by continuously integrating belt speed input. It also maintains correct indices of head and tail cells in a rolling storage queue explained next.
2) *Package Data Store:* Stores package sizes and arrival odometer values within a queue. The queue is implemented as a rolling array. When a new package arrives, its size and the current odometer reading of the belt is recorded in the tail cell.
3) *Package Release Controller:* Continuously calculates position of head package on belt. When the rear edge of the head package falls past the end of the belt, it is removed from the queue and passed via a pulse to the downstream machine.
4) *Upstream and Downstream Sensor Controllers:* Continuously calculate positions of the head and tail packages. Whenever any portion of a package is within an endzone, the endzone controller generates a sensor value of 1; otherwise, controllers generate sensor values of 0 when an endzone is empty.
5) *Transfer and Receive Counters:* Maintain counts of transferred and received packages

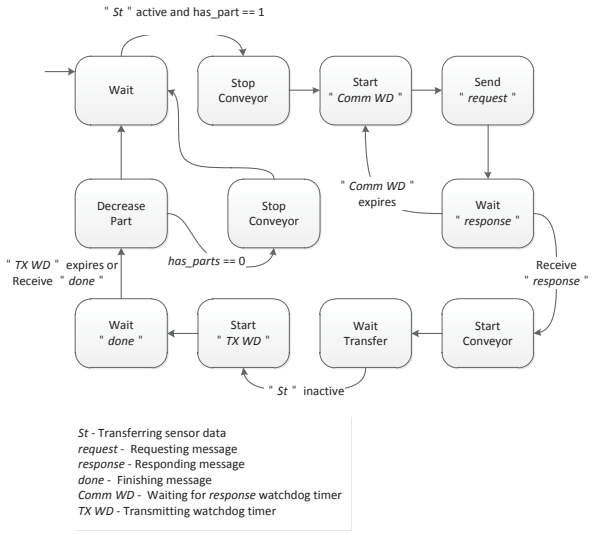

Fig. 10. Turnaround Transfer State Machine

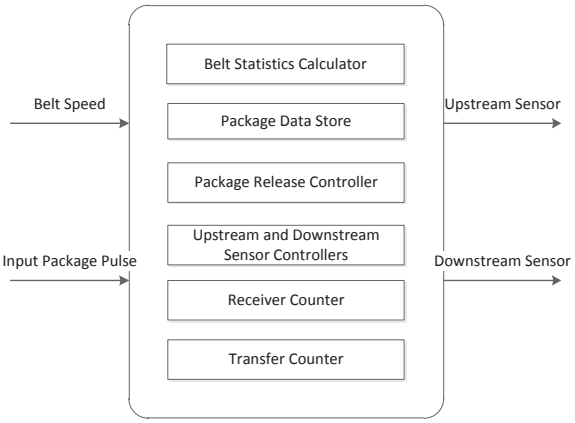

Fig. 11. Simulation of the Physics of a Segment

Figure 12 depicts a high level perspective of the physics of simulated turnaround. It comprises the following building blocks:

1) *Package Position Calculator:* Continuously integrates motor control signals for East-West and North-South axes in order to calculate and maintain position of package on belt.

2) *Package Change Controller:* Detects when packages enter or leave the turnaround machine. Whenever a new package arrives (via pulse), this controller resets the Package Position Controller to the position and size of the newly arrived package. Note that if a package already exists on the belt when a new one arrives, the old package will be lost. When a package is determined to have moved completely off of the machine, a package pulse is generated on the appropriate directional output and the Package Position Controller is reset with a null package.
3) *Edge Sensor Controller:* Maintains sensor outputs for Edge Sensor Beams on all four sides of machine. Whenever any portion of a package is determined to reside within an edge beam, the appropriate edge sensor outputs the size of the package breaking the beam. Outputs for unbroken edge beams are set at zero.

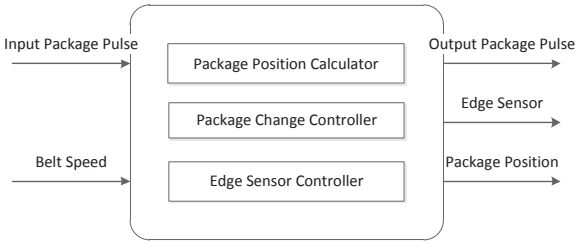

Fig. 12. Simulation of the Physics of a Turnaround

V. Experimental Results

The experimental results we describe in this section are the results of analysis we collected for an example reconfigurable conveyor system layout modeled in Figure 5. Note that the maximum sustainable rates at input bins and observed trhoughput at output bins can be affected by various cyber and physical parameters, such as length and velocity of belts, coverage of sensors, and watchdog timer values used in software controllers.

Table V shows the configured variables for the experiment; every Segment and Turnaround is identically set up with the parameters in the table. Every type of package (small, medium, and large) is produced evenly at input bins by random functions according to a uniform distribution. The length of each package is fixed: small packages are 1m, medium packages are 1.5m, and large packages are 2m. The interval space between packages can vary depending on traffic patterns of a system, but it is primarily affected by coverage of the sensors.

TABLE I
Experiment Parameters

Block Type	Belt Velocity	Belt Length	Sensor Zone	RX WD	TX WD	COMM WD
Segment	1 m/s	20 m	2 m	10 secs	10 secs	0.5 secs
Turnaround	1 m/s	5 m	1 m	10 secs	10 secs	0.5 secs

In case of a system configured by the table above, the coverage of sensors for a Segment is 2m and for Turnaround is 1m, so spacing between packages is usually 4m when a package is transferred between Segments and 3m when a package is transferred between a Segment and a Turnaround.

Figure 13 shows input rates of each bin and the rate of the total packages generated. The graph shows that the rate of total numbers of packages generated rapidly increases until the system saturates. Input bins periodically send a request to the next Segment every 0.5 sec. Over time, the next Segment reaches its maximum sustainable limit and the rates of input bins are stabilized through back pressure (*i.e.*, flow control).

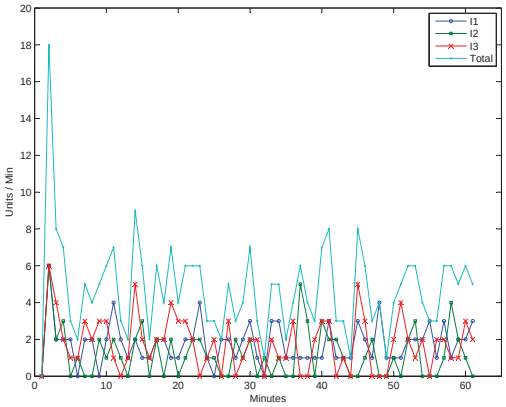

Fig. 13. Sustained Input Rates of Packages

The throughput observed at each output bin and the rate of total numbers of packages arrived are shown in Figure 14. Comparing the numbers to the input rates, the output rate of total numbers of packages arrived are already stabilized since the rate of arrivals are balanced as the packages move through the different parts of the conveyor system.

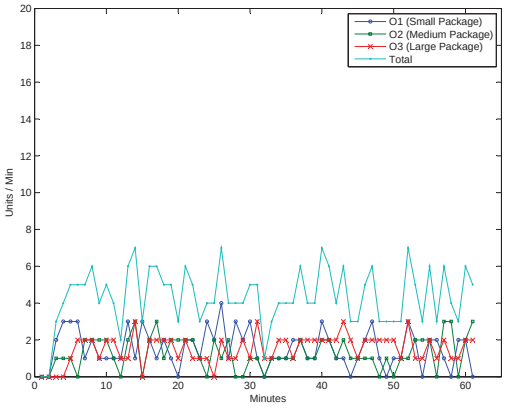

Fig. 14. Observed Output Rates of Packages

We conducted another experiment changing COMM_WD values in transfer state machines to investigate how throughput

TABLE II
NUMBER OF PACKAGES ARRIVED FOR DIFFERENT COMM_WD TIMER VALUES

COMM_WD	Small Packages Arrived	Medium Packages Arrived	Large Packages Arrived
0.5sec	94	94	78
0.1sec	102	79	62

of a system is changed by altering a watchdog timer value. If the numbers of packages arrived at output bins according to types of packages is not even, it indicates there is starvation of some types of packages in routes. The outcomes shown in Figure 15 demonstrate that if the number of packages of each bin is balanced and starvation in a system is infrequent, total throughput is higher.

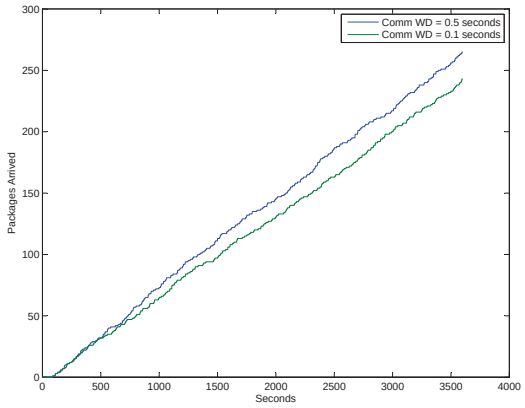

Fig. 15. Comparing the Total Numbers of Packages Arrived

When Comm_WD is 0.5sec, large packages were starved. When Comm_WD is 0.1sec, it shows that both medium and small packages were starved. Table II illustrates the outcomes. Therefore, the performance of a system when Comm_WD is 0.5sec is better than the one of a system when Comm_WD is 0.1sec because it has less starvation in the paths.

VI. CONCLUDING REMARKS

This paper presented a model-driven analysis framework for collecting different performance metrics for reconfigurable conveyor systems. A model-driven analysis framework makes it possible to decouple the activity of describing the topology of the proposed conveyor system from the analysis engine that collects different metrics to evaluate the properties of the topology. This separation enables the model-driven framework to change the underlying analysis engine while also enabling the generative mechanisms to synthesize code artifacts when the system is actually fielded. As a result complete automation can be realized using a common framework.

Our research in developing such a capability, particularly, the underlying analysis engine for a CPS system such as reconfigurable conveyors illustrates adverse consequences of certain design decisions. For example, principles such as separation of concerns which are highly effective in software designs (*i.e.*, a cyber world issue) tend to produce incorrect results due to the complex interaction of timing issues and concurrent behavior of physical systems. Moreover, we observed that concurrent behavior in the cyber world must be correctly synchronized because these cyber artifacts often share common physical resources. For example, the receiver and transfer finite state machines share a common belt and hence their behaviors must be synchronized else the system may result in unforeseen consequences.

In the same manner, trying to arbitrarily manipulate physical-level parameters, such as reducing the inter package spacing to maximize throughput may also lead to adverse consequences. In our case, we also observed that long periods of starvation resulted for certain paths in the topology resulting from such a physical-level manipulation and how it impacted the behavior of the concurrent finite state machines and the timers.

Our future work in this area will explore analysis of failures in the system. We plan to target both the physical failures, such as motor failing, and cyber failures, such as the microcontroller logic failing. Our goal is to identify the impact on the system throughput due to failures, and also to understand how runtime adaptation by rerouting goods will help to maintain acceptable levels of performance in system operation. We also plan to prioritize according to package types.

ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation NSF SHF/CNS Award CNS 0915976, and Vanderbilt Discovery Grant. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or Vanderbilt University.

REFERENCES

[1] Dynamic Conveyor Corporation, "Reconfigurable Modular Conveyors: Equipment that Unites Controllers and Engineers," Online Article in Medical Design Technology (MDT) Magazine, Jun. 2010.

[2] G. Moreno and P. Merson, "Model-driven performance analysis," *Quality of Software Architectures. Models and Architectures*, pp. 135–151, 2008.

[3] W. Wolf, "Cyber-Physical Systems," *Computer*, vol. 42, no. 3, pp. 88–89, 2009.

[4] G. Karsai and J. Sztipanovits, "Model-Integrated Development of Cyber-Physical Systems," in *Software Technologies for Embedded and Ubiquitous Systems*, ser. Lecture Notes in Computer Science, U. Brinkschulte, T. Givargis, and S. Russo, Eds. Springer Berlin / Heidelberg, 2008, vol. 5287, pp. 46–54.

[5] M. Mernik, J. Heering, and A. M. Sloane, "When and How to Develop Domain-specific Languages," *ACM Computing Surveys*, vol. 37, no. 4, pp. 316–344, 2005.

[6] E. Lee, "Cyber Physical Systems: Design Challenges," in *11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing (ISORC), 2008.* Orlando, FL: IEEE, 2008, pp. 363–369.

[7] ——, "Computing Needs Time," *Communications of the ACM*, vol. 52, no. 5, pp. 70–79, 2009.

[8] K. Czarnecki and U. Eisenecker, *Generative Programming: Methods, Tools, and Applications*. Boston: Addison-Wesley, 2000.

[9] F. de Lamotte, P. Berruet, and J.-L. Philippe, "Evaluation of Reconfigurable Manufacturing Systems Configurations using Tolerance Criteria," in *IEEE 32nd Annual Conference on Industrial Electronics, IECON 2006*, Paris, France, Nov. 2006, pp. 3715 –3720.

[10] D. C. Schmidt, "Model-Driven Engineering," *IEEE Computer*, vol. 39, no. 2, pp. 25–31, 2006.

[11] S. Heragu, G. Meng, W. Zijm, and J. van Ommeren, "Design and Analysis of Reconfigurable Layout Systems," Department of Applied Mathematics, University of Twente, Memorandum no. 1604, Tech. Rep., 2001.

[12] W. Li, H. Yang, and T. Murata, "An Expandable Petri Net Framework for Method Behavior Evaluation of Reconfigured Equipment," in *Proceedings of the International MultiConference of Engineers and Computer Scientists (IMECS 2010)*, Mar. 2010.

[13] A. M. Farid and D. C. McFarlane, "Production degrees of freedom as manufacturing system reconfiguration potential measures," *Journal of Engineering Manufacture*, vol. 222, no. 10, pp. 1301–1314, 2008.

[14] J. Heilala and P. Voho, "Modular Reconfigurable Flexible Final Assembly Systems," *Assembly Automation*, vol. 21, no. 1, pp. 20–30, 2001.

[15] D. Kalita and P. Khargonekar, "Formal Verification for Analysis and Design of Logic Controllers for Reconfigurable Machining Systems," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 4, pp. 463–474, Aug. 2002.

[16] J. Li, X. Dai, and Z. Meng, "Improved Net Rewriting system-based Approach to Model Reconfiguration of Reconfigurable Manufacturing Systems," *The International Journal of Advanced Manufacturing Technology*, vol. 37, pp. 1168–1189, 2008, 10.1007/s00170-007-1037-5. [Online]. Available: http://dx.doi.org/10.1007/s00170-007-1037-5

[17] S. Kolla, J. Michaloski, and W. Rippey, "Evaluation of Component-based Reconfigurable Machine Controllers," in *Proceedings of the 5th Biannual World Automation Congress, 2002*, vol. 14, Jun. 2002, pp. 625 – 630.

[18] S. Kuruvilla, S. Gokhale, and S. Sastry, "Reliability Evaluation of Reconfigurable Conveyor Systems," in *IEEE International Conference on Automation Science and Engineering (CASE 2008)*. IEEE, 2008, pp. 929–934.

[19] K. K. Mamidisetty, M. Duan, S. Sastry, and P. S. Sastry, "Multipath Dissemination in Regular Mesh Topologies," *IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS*, vol. 20, no. 8, 2009.

[20] B. Archer and S. Sastry, "Worst-case End-to-end Response Time Analysis for Composable Conveyor Systems," in *Work-in-progress Session of the 22nd Euromicro Conference on Real-time Systems (ECRTS 2010)*, R. I. Davis, Ed., Brussels, Belgium, Jul. 2010, pp. 49–52.

[21] B. Archer, S. Sastry, A. Rowe, and R. Rajkumar, "Profiling Primitives of Networked Embedded Automation," in *IEEE International Conference on Automation Science and Engineering (CASE), 2009*. Bangalore, India: IEEE, 2009, pp. 531–536.

[22] Á. Lédeczi, Á. Bakay, M. Maróti, P. Völgyesi, G. Nordstrom, J. Sprinkle, and G. Karsai, "Composing Domain-Specific Design Environments," *Computer*, vol. 34, no. 11, pp. 44–51, 2001.