MODELING AN IMPACT CONTROL STRATEGY USING HYPA

Pieter J.L. Cuijpers [*,1] **Michel A. Reniers** [*]

[*] *Technische Universiteit Eindhoven (TU/e), Eindhoven*

Abstract: We analyze a control strategy for the pick-and-place module of a component mounting device. We use a combination of techniques from bondgraph-theory, systems theory, process algebra and differential algebra to achieve this, and we show how the hybrid process algebra HyPA aides us in combining these techniques and in using them on a common hybrid model of the device.

Keywords: hybrid systems, process algebra, bondgraph theory, impact control

1. INTRODUCTION

In this paper, we describe and analyze a control strategy for the pick-and-place module of a component mounting device, see figure 1, using the hybrid process algebra HyPA (Cuijpers and Reniers, 2005). The objective of this control strategy, designed by Philips CFT and Assembleon, is to bring a component to a PCB (Printed Circuit Board) as quickly as possible, and press it onto the PCB with sufficient force to make it stick. This all should be done without damaging the component. The focus of our analysis, is to show under which conditions safety of the controller can be guaranteed. In other words, we aim to find conditions under which it is certain that the component is not damaged.

The modeling and analysis of the pick-and-place module is carried out in the hybrid process algebra HyPA. This process algebra focuses on the description of so-called hybrid systems, i.e. on models of systems in which both continuous and discrete behavior occur. In the case of the pick-and-place module, hybrid modeling turns out to be useful, because it allows us to abstract from the precise mechanics of impact and other relatively fast behavior. This makes the dynamic behavior of the system much simpler which, rather unexpectedly, allows us to find analytic bounds for safety for this particular case study.

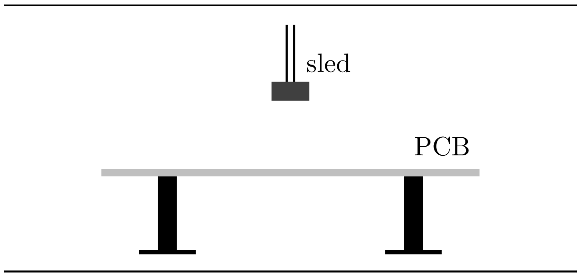

Fig. 1. A pick-and-place module

Apart from providing the user with a framework to specify hybrid systems, HyPA also allows algebraic reasoning about the compositional structure of such systems, i.e. on the way in which a hybrid system is composed from subsystems. One of the strengths of HyPA, is in finding different abstract representations of a system. In this paper, we first specify the system as a parallel composition of more-or-less standard components from bondgraph theory (Karnopp *et al.*, 1990; Mosterman *et al.*, 1998; Cuijpers *et al.*, 2004), and then rewrite it into a semi-linear representation (Usenko, 2002; van de Brand *et al.*, 2006) that

[1] We would like to express our gratitude to Progress/STW (Grant EES5173), Philips CFT and Assembleon, for their financial and material support of this case study, and of our project on the development of hybrid systems theory.

gives more insight in the functional behavior of the system. The exact functional behavior is extracted from this last representation using analysis of the differential equations and difference equations that occur in it.

In a sense, the focus of the process algebraic approach on compositional structure is supplemental to the kind of analysis of hybrid behavior as described in, for example, (Mosterman *et al.*, 1998; Febbraro *et al.*, 2001; Alur and Dill, 1994; Heemels *et al.*, 2001; Henzinger, 1996; van der Schaft and Schumacher, 2000). These latter approaches often consider a hybrid model that is already written in a specific form, and are able to give a very detailed analysis of such a model. The process algebraic approach provides a way of switching between different compositional representations of the model, i.e. between the different forms necessary for certain kinds of analysis. Recently, a number of other process algebras for hybrid systems have been introduced as well (Rounds and Song, 2003; Bergstra and Middelburg, 2005; van Beek *et al.*, 2006), which differ on a number of subtle details that are outside the scope of this paper. We suppose that these process algebras are equally suitable for the purpose of modeling and analyzing a system as presented in this paper, but it may be interesting to verify this in the future.

The structure of this paper is as follows. In section 2, we briefly introduce the hybrid process algebra HyPA. In section 3, we discuss the central model for this case study, namely that of the pick-and-place module. In section 4, we extend it with a model of a measuring and control strategy proposed by Philips CFT (Mateboer, 1999). In section 5, we formalize the analysis goal of this paper, i.e. we formalize what we mean by safety of a system. Finally, in section 6, we discuss the results of the actual analysis of the controlled pick-and-place module, and in section 7, we give some conclusive remarks and recommendations for future work.

2. HYBRID PROCESS ALGEBRA

In this section, the part of the syntax of HyPA that is used in this paper is discussed. The discussion presented here is adapted from (van de Brand *et al.*, 2006). For a more detailed work on this algebra see (Cuijpers and Reniers, 2005; Cuijpers, 2004).

The syntax of HyPA is an extension of the process algebra ACP (Baeten and Weijland, 1990), with the disrupt operator from LOTOS (Brinksma, 1985) and with variants of the flow clauses and re-initialization clauses from the event-flow formalism introduced in (van der Schaft and Schumacher, 2000). The signature of HyPA contains the following constant and function symbols:

(1) discrete actions $a \in \mathcal{A}$,
(2) flow clauses $c \in C$,
(3) a family of process re-initialization operators $d \gg \_$ where $d \in D$,
(4) alternative composition $\_ \oplus \_$,
(5) sequential composition $\_ \odot \_$,
(6) disrupt $\_ \blacktriangleright \_$ and left-disrupt $\_ \rhd \_$,
(7) parallel composition $\_ \parallel \_$,
(8) a family of encapsulation operators $\partial_H(\_)$, where $H \subseteq \mathcal{A}$, and predicate encapsulation operators $\partial_{P_m}(\_)$, where $P_m \in \mathcal{P}_m$.

The binding order of these operators is as follows: $\odot$, $\blacktriangleright$, $\rhd$, $d \gg$, $\parallel$, $\oplus$, where sequential composition binds strongest and alternative composition binds weakest. These constants and operators are described informally below.

Flow clauses are used to model continuous, never terminating, physical behavior by describing how the model variables $\mathcal{V}_\mathrm{m}$ are allowed to change through time. A flow clause is a pair $(V \,|\, P_f)$ of a set of model variables $V \subseteq \mathcal{V}_\mathrm{m}$ and a flow predicate $P_f \in \mathcal{P}_f$. This flow predicate may, for example, contain differential equations and algebraic equations. The set V models which variables are *not* allowed to jump at the beginning of a flow.

A process re-initialization $d \gg p$ models the behavior of p where the model variables are submitted to a discontinuous change as specified by the re-initialization clause d. A re-initialization clause describes a set of *re-initializations*, where a re-initialization is a pair of valuations representing the values of the model variables prior to and immediately after the re-initialization. The set of all re-initializations $Val \times Val$ is denoted $\mathcal{R}$.

A re-initialization clause is a pair $[V \,|\, P_r]$ of a set of model variables $V \subseteq \mathcal{V}_\mathrm{m}$ and a re-initialization predicate $P_r \in \mathcal{P}_r$. The set V models which variables are allowed to change. Note that this is precisely opposite to flow clauses, where V denotes those variables that do *not* change (initially). Predicate P_r models the discontinuous changes. In a predicate, x^- denotes the value of a variable x before re-initialization, x^+ denotes the value of a variable x after re-initialization, and x' denotes the difference between x^+ and x^-, i.e., $x' = x^+ - x^-$. Also, P_r^- denotes the predicate P_r with all occurrences of x replaced by x^-.

The alternative composition $p \oplus q$ models a (non-deterministic) choice between the processes p and q. The sequential composition $p \odot q$ models a sequential execution of processes p and q. The process q is executed after (successful) termination of the process p.

The disrupt $p \blacktriangleright q$ models a kind of sequential composition where the process q may take over execution from process p at any moment, without waiting for its termination. This composition is essential for modeling two flow clauses executing one after the other, since the behavior of flow clauses never terminates. The left-disrupt $p \rhd q$ first executes a part of the process p and then behaves as a normal disrupt.

The parallel composition $p \parallel q$ models concurrent execution of p and q. The intuition behind this concurrent execution is that discrete actions are executed in an interleaving manner, with the possibility of synchronization of actions (as in ACP, where synchronization is called communication), while flow clauses are forced to synchronize, and can only synchronize if they accept the same solutions. The synchronization of actions takes place using a (partial, commutative and associative) communication function $| \in \mathcal{A} \times \mathcal{A} \mapsto \mathcal{A}$. For example, if the actions a and a' synchronize, then the resulting action is $a'' = a \,|\, a'$. Actions cannot synchronize with flow clauses, and in a parallel composition between those, the action executes first. Re-initializations synchronize only if the processes on which they act synchronize.

Encapsulation $\partial_H(p)$ models that the discrete actions from the set $H \subseteq \mathcal{A}$ are blocked during the execution of process p. This operator is often used in combination with the parallel composition to model that synchronization between discrete actions is enforced. Predicate encapsulation $\partial_{P_m}(p)$ models that all actions and flows that (at some point in time) satisfy the predicate on model variables P_m, are blocked. In (Cuijpers, 2004) and the present paper, it is used to formalize and analyze safety requirements on processes.

Terms can be constructed using variables from a given set of process variables $\mathcal{V}_p$ (with $\mathcal{V}_p \cap \mathcal{V}_m = \emptyset$), as usual. Finally, all processes should be interpreted in the light of a set E of recursive definitions of the form $X : p$, where X is a process variable and p is a term.

For some examples of modeling using HyPA we refer to (Cuijpers *et al.*, 2004; Cuijpers, 2004; Man *et al.*, 2005).

3. THE PICK-AND-PLACE MODULE

In this section, we give a model for the pick-and-place module based on bondgraph modeling.

In (Mateboer, 1999), a model of the pick-and-place module is used that contains differential and algebraic equations for the collision mechanics. Simulations are performed, to see how changes in the characteristics of the PCB influence the

$$\begin{tikzcd}
\text{Mass}(m_s) & \text{Junction} \arrow[l, harpoon', "1"'] \arrow[d, harpoon', "3"'] & \text{Actuator}(F) \arrow[l, harpoon', "2"'] \\
& \text{Collision}(\text{Act},\text{InAct}) \arrow[d, harpoon', "4"'] & \\
\text{Mass}(m_p) & \text{Junction} \arrow[l, harpoon', "5"'] \arrow[r, harpoon', "6"] \arrow[d, harpoon', "7"] & \text{Spring}(k_p) \\
& \text{Damper}(b_p) &
\end{tikzcd}$$

Fig. 2. Bondgraph of the pick-and-place module

performance of the controller. One of the conclusions of that report is that, if the characteristics of the collision mechanics are dominant over the characteristics of the PCB (an assumption that is reasonable in practice), then the impact behavior has ended before it is detected.

This relatively fast impact behavior, is the first reason why we abstract from the precise impact mechanics in our model, replacing it by discontinuous behavior. The second reason is that the parameters of the collision may vary wildly as different components and PCBs are used. Abstraction from the precise mechanics, means that we are robust against those variations.

In figure 2, a bondgraph model (Karnopp *et al.*, 1990) of the impact module is depicted. A bondgraph gives a graphical representation of the energy exchange in a physical system. In our example, the components of which the bondgraph consists are a mass (m_s) that models the sled, a force (F) that models the actuator through which the sled is controlled, a mass-spring-damper system (m_p, k_p, b_p) modeling the flexible PCB, and junctions (including a special collision junction) that model the exchange of energy between the other components.

Classically, the semantics of a bondgraph is given by a set of differential and algebraic (so called *constitutive*) equations for each of the bondgraph components (Karnopp *et al.*, 1990). However, inconsistencies tend to arise in these equations whenever connections between components are made or broken, which is what happens during collisions. In (Mosterman *et al.*, 1998), a method was proposed to deal with these inconsistencies while performing simulations of a system, and switching junctions such as our collision component were introduced to model the making and breaking of connections. In (Cuijpers *et al.*, 2004; Cuijpers, 2004) an algebraic semantics for hybrid bondgraphs was given using so called *hybrid constitutive processes* in the process algebra HyPA. These constitutive processes contain the usual constitutive differential and algebraic equations for continuous behavior, and in addi-

Table 1. Physical meaning of the variables (and constants).

Variable	Physical meaning	Unit
a	acceleration	$\frac{m}{s^2}$
b	damping constant	$\frac{Ns}{m} = \frac{kg}{s}$
E	energy	$J = \frac{kgm^2}{s^2}$
F	force	$N = \frac{kgm}{s^2}$
k	elasticity / spring constant	$\frac{N}{m} = \frac{kg}{s^2}$
m	mass	kg
p	momentum	$\frac{kgm}{s}$
v	velocity	$\frac{m}{s}$
x	position	m

tion contain constitutive difference equations that describe the discontinuities in behavior due to the aforementioned inconsistencies. Thus, the semantics of a hybrid bondgraph is formed by a parallel composition of constitutive hybrid processes for each of the bondgraph components. The constitutive hybrid processes associated with our impact system are explained below, and summarized in table 2.

The classical constitutive equations of bondgraphs are based on the insight that in many physical systems energy exchange (i.e. *power*) between two components is the product of two variables called *generalized effort* and *generalized flow*. In turn, generalized effort and generalized flow are the time-derivatives of the two state-variables of a physical component, which are *generalized momentum* and *generalized displacement* respectively. In mechanical systems, such as the one under study, *energy* (E) is a combination of kinetic and potential energy, generalized effort is better known as *force* (F), generalized flow is called *velocity* (v), generalized momentum is called *momentum* (p) and generalized displacement is called *position* (x). The behavior of each mechanical component can be expressed using these five variables, and their fundamental relations ($\dot{E} = F \cdot v$, $\dot{x} = v$ and $\dot{p} = F$) are captured in the Bond process in table 2. Note, that we have adapted the notation in this paper to that of the mechanical domain (see table 1).

In figure 2, each of the bonds (i.e. the half-arrows) is identified by a number. With each bond i variables E_i, p_i, and x_i, F_i and v_i, representing energy, momentum, position, force and velocity, of that bond, are associated. The direction of the bonds defines the positive direction of the flow of energy. The positive direction for the physical variables (e.g. which direction of movement is considered a 'positive' velocity) should be chosen in accordance to this. In relation to figure 1, all 'upward' displacements, velocities and forces are considered positive, contrary to what one might intuitively expect.

The continuous behavior of the mechanical components is described using the usual constitutive equations for conservation of energy and momentum. The components are additionally described by the equation $F = m \cdot a$ for a mass, $F = b \cdot v$ for a damper and $F = k \cdot x$ for a spring. The Actuator(F) process acts as a source of a constant force F. The discontinuous behavior of these mechanical components can be derived using partial integration from these laws. Partial integration of energy E as a function of momentum p, for example, gives us that the change in energy E' depends on the momentum p^- prior to the change and the momentum p^+ after the change, satisfying $E' = \frac{(p^+)^2 - (p^-)^2}{2 \cdot m}$. See (Cuijpers *et al.*, 2004) for a more complete treatment of the theory behind these derivations.

The Junction process serves to describe preservation of momentum and energy. Note our slight abuse of notation regarding the direction in which variables are to be interpreted. Firstly, for a set C of bonds, V_C denotes the set $\{E_c, p_c, x_c \mid c \in C\}$ of state-variables. Secondly, whenever we have an inward (outward) half-arrow in the bondgraph, we write a corresponding $+$ $(-)$ sign in the set C to signify the direction of the bond, and we write a $+$ $(-)$ in the corresponding constitutive equations whenever a $\pm$ sign occurs in these equations (typically, it will occur in front of a F_c or v_c).

The Collision process describes how junctions are made and/or broken as a result of collisions between masses (in (Cuijpers *et al.*, 2004), the Collision process is called a $(1/E)$-junction). The predicates Act and InAct describe the situation that the two masses are connected and act as one mass and the situation where the two masses are not connected, respectively (see table 2). Indeed, these two predicates are not local, in the sense that they use variables of components that are not directly connected by bonds. From a bondgraph perspective, this is still correct, since there is no exchange of energy defined through Act and InAct. However, the occurrence of non-local variables in the definition of Collision is, admittedly, confusing. A more elaborate variant of the bondgraph formalism exists that takes such exchange of information into account as well. It is for example used in (Mosterman *et al.*, 1998), but it is beyond the scope of this paper to treat such variants here.

4. CONTROL STRATEGY

In this section, we discuss the control strategy that is applied to the pick-and-place module. We base our discussion on the control strategy as it was first suggested by Philips CFT (Mateboer, 1999).

The control strategy consists of two phases. In the first phase, the sled is brought down using

Table 2. Definitions for the constitutive hybrid processes.

$$
\begin{array}{rcl}
\text{Module} & : & \text{Mass}_1(m_\text{s}) \parallel \text{Bond}_1 \parallel \text{Actuator}_2(F) \parallel \text{Bond}_2 \parallel \text{Bond}_3 \parallel \text{Bond}_4 \parallel \text{Mass}_5(m_\text{p}) \parallel \text{Bond}_5 \parallel \text{Spring}_6(k_\text{p}) \\
 & \parallel & \text{Bond}_6 \parallel \text{Damper}_7(b_\text{p}) \parallel \text{Bond}_7 \parallel \text{Junction}_{\{-1,2,-3\}} \parallel \text{Collision}_{\{3,-4\}} \parallel \text{Junction}_{\{4,-5,-6,-7\}} \\
\text{Mass}_i(m) & : & \left(\left[E_i, p_i, x_i \;\middle|\; E_i' = \frac{(p_i^+)^2 - (p_i^-)^2}{2 \cdot m} \wedge x_i' = 0 \right] \gg \left(E_i, p_i, x_i \;\middle|\; p_i = m \cdot v_i \right) \right) \blacktriangleright \text{Mass}_i(m) \\
\text{Spring}_i(k) & : & \left(\left[E_i, p_i, x_i \;\middle|\; E_i' = k \cdot \frac{(x_i^+)^2 - (x_i^-)^2}{2} \wedge p_i' = 0 \right] \gg \left(E_i, p_i, x_i \;\middle|\; x_i = \frac{1}{k} \cdot F_i \right) \right) \blacktriangleright \text{Spring}_i(k) \\
\text{Damper}_i(b) & : & \left(\left[E_i, p_i, x_i \;\middle|\; E_i' = 0 \wedge p_i' = 0 \wedge x_i' = 0 \right] \gg \left(E_i, p_i, x_i \;\middle|\; F_i = b \cdot v_i \right) \right) \blacktriangleright \text{Damper}_i(b) \\
\text{Actuator}_i(F) & : & \left(\left[E_i, p_i, x_i \;\middle|\; E_i' = 0 \wedge p_i' = 0 \right] \gg \left(E_i, p_i, x_i \;\middle|\; F_i = F \right) \right) \blacktriangleright \text{Actuator}_i(F) \\
\text{Bond}_i & : & \left(\left[E_i, p_i, x_i \;\middle|\; \mathit{true} \right] \gg \left(E_i, p_i, x_i \;\middle|\; \dot{E}_i = F_i \cdot v_i \wedge \dot{p}_i = F_i \wedge \dot{x}_i = v_i \right) \right) \blacktriangleright \text{Bond}_i \\
\text{Junction}_C & : & \left(\left[V_C \;\middle|\; \begin{array}{l} \sum_{c \in C} \pm E_c' = 0 \wedge \sum_{c \in C} \pm p_c' = 0 \\ \forall_{c,c' \in C}\; x_c' = x_{c'}' \end{array} \right] \gg \left(V_C \;\middle|\; \begin{array}{l} \sum_{c \in C} \pm F_c = 0 \\ \forall_{c,c' \in C}\; v_c = v_{c'} \end{array} \right) \right) \blacktriangleright \text{Junction}_C \\
\text{Collision}_C & : & \left(\left[V_C \;\middle|\; \begin{array}{l} \text{Act}^- \wedge \sum_{c \in C} \pm E_c' = 0 \\ \sum_{c \in C} \pm p_c' = 0 \wedge \forall_{c,c' \in C}\; x_c' = x_{c'}' \end{array} \right] \gg \left(V_C \;\middle|\; \begin{array}{l} \text{Act} \wedge \sum_{c \in C} \pm F_c = 0 \\ \forall_{c,c' \in C}\; v_c = v_{c'} \end{array} \right) \right. \\
 & \oplus & \left[V_C \;\middle|\; \text{InAct}^- \wedge \forall_{c \in C}\; E_c' = 0 \wedge x_c' = 0 \right] \gg \left(V_C \;\middle|\; \text{InAct} \wedge \forall_{c \in C}\; F_c = 0 \right) \\
 & \oplus & \left[V_C \;\middle|\; \begin{array}{l} \text{InAct}^- \wedge \sum_{c \in C} \pm E_c' \geq 0 \\ \sum_{c \in C} \pm p_c' = 0 \wedge \forall_{c,c' \in C}\; x_c' = x_{c'}' \end{array} \right] \gg \left(V_C \;\middle|\; \begin{array}{l} \text{Act} \wedge \sum_{c \in C} \pm F_c = 0 \\ \forall_{c,c' \in C}\; v_c = v_{c'} \end{array} \right) \\
 & \oplus & \left[V_C \;\middle|\; \begin{array}{l} \text{Act}^- \wedge \sum_{c \in C} \pm E_c' \geq 0 \\ \sum_{c \in C} \pm p_c' = 0 \wedge \forall_{c,c' \in C}\; x_c' = x_{c'}' \end{array} \right] \gg \left(V_C \;\middle|\; \text{InAct} \wedge \forall_{c \in C}\; F_c = 0 \right) \\
 & \oplus & \left[V_C \;\middle|\; \text{InAct}^- \wedge \forall_{c \in C}\; \pm E_c' \geq 0 \wedge x_c' = 0 \right] \gg \left(V_C \;\middle|\; \text{Act} \wedge \sum_{c \in C} \pm F_c = 0 \wedge \forall_{c,c' \in C}\; v_c = v_{c'} \right) \\
 & \oplus & \left. \left[V_C \;\middle|\; \text{Act}^- \wedge \forall_{c \in C}\; \pm E_c' \geq 0 \wedge x_c' = 0 \right] \gg \left(V_C \;\middle|\; \text{InAct} \wedge \forall_{c \in C}\; F_c = 0 \right) \right) \blacktriangleright \text{Collision}_C \\
\text{Act} & \equiv & (x_1 = x_5 \wedge v_1 > v_5) \vee (x_1 = x_5 \wedge v_1 = v_5 \wedge \frac{1}{m_\text{s}} \cdot (F_1 + F_3) \geq \frac{1}{m_\text{p}} \cdot (F_2 - F_4)) \\
\text{InAct} & \equiv & x_1 \leq x_5
\end{array}
$$

Table 3. Definitions for the controller, the sensor and the system.

$$
\begin{array}{rcl}
\text{System} & : & \left[x_5^- = v_5^- = 0 \wedge \mathit{margin} \wedge \mathit{consistent} \right] \gg (\text{Module} \parallel \text{Controller} \parallel \text{Sensor}) \\
\text{Controller} & : & \left(\begin{array}{l} -F_\text{sat} \leq F = -K \cdot (v_1 + v_\text{seek}) \leq F_\text{sat} \\ \vee \\ -F_\text{sat} \geq -K \cdot (v_1 + v_\text{seek}) \wedge F = -F_\text{sat} \\ \vee \\ -K \cdot (v_1 + v_\text{seek}) \geq F_\text{sat} \wedge F = 0 \end{array} \right) \blacktriangleright \text{impact?} \odot \left(F = -F_\text{sat} \right) \\
\text{Sensor} & : & \left(\mathit{margin} \right) \blacktriangleright \left[\begin{array}{c} v_1 \\ \mathit{clck} \end{array} \;\middle|\; \begin{array}{l} \mathit{margin}^- \\ \neg \mathit{margin}^+ \\ \mathit{clck}^+ = 0 \end{array} \right] \gg \left(\begin{array}{c} v_1 \\ \mathit{clck} \end{array} \;\middle|\; \begin{array}{l} \dot{\mathit{clck}} = 1 \\ \mathit{clck} \leq t_\text{detect} \end{array} \right) \blacktriangleright \left[0 < \mathit{clck}^- \leq t_\text{detect} \right] \gg \text{impact!} \\
\mathit{margin} & \equiv & -v_\text{seek} - v_\text{detect} \leq v_1 \leq -v_\text{seek} + v_\text{detect} \\
\mathit{consistent} & \equiv & \left(\begin{array}{l} p_1 = m_\text{s} \cdot v_1 \wedge p_5 = m_\text{p} \cdot v_5 \wedge x_6 = \frac{1}{k} \cdot F_6 \wedge F_2 = F \wedge F_7 = b \cdot v_7 \\ F_2 - F_1 = F_3 = F_4 = F_5 + F_6 + F_7 \wedge v_1 = v_2 = v_3 \wedge v_4 = v_5 = v_6 = v_7 \end{array} \right)
\end{array}
$$

a proportional feedback law or P-controller (Dorf and Bishop, 1995). This means, that the controller measures the velocity v_1 of the sled and attempts to keep it at a constant value v_seek. It does this by changing the applied force F proportionally, by a constant factor K, to the difference between v_1 and v_seek. Note, that the controller synchronizes the values of v_1 and F with the sled and actuator components, respectively. In the second phase, the sled is pushed onto the PCB with a constant force, by bringing the force-actuator into saturation (F_sat). Naturally, in the actual system, the sled is brought up after placing the component on the PCB, but this is outside the scope of our studies at the moment. Here, we focus on one placement only. Our two phase control strategy is captured in the process Controller in table 3.

The controller switches from the first phase to the second when an impact is detected (the action impact?). A logical first step to detect this impact is by measuring the distance between sled and PCB. However, this approach needs an additional sensor, and turned out to be to expensive to be used in the final product. Instead, impact detection (the action impact!) is carried out by detecting an abrupt change in the velocity of the sled. The drawback of using a velocity sensor for impact detection, is that the detection is never immediate. We need to allow some time t_detect between impact and detection, and use an internal clock *clck* in our model, to measure this time. In order to make the model of the sensor more realistic, we also include a detection margin (with $v_{\mathrm{detect}} > 0$) on the velocity (see the process Sensor in table 3). The sensor and the controller communicate over a channel according to impact! | impact? = impact.

For the impact detection to work correctly, it is important that the PCB is initially at rest $x_5 = v_5 = 0$. Furthermore, in order for an impact to be detectable, v_1 should be higher than a certain threshold. Lastly, for technical reasons, the initial state of the system should be physically consistent. The initial value of the physical variables should be chosen such that there is a possible solution to the constitutive equations. This is captured in the predicate *consistent*. In the remainder of this paper, we call the system as a whole System (see table 3).

5. SAFETY REQUIREMENTS

In (Mateboer, 1999) it is assumed that the placed component will be damaged when there is too great a force ($F_{\max}$) acting on it. Using that assumption, a maximum impact velocity $v_{\max} = \frac{F_{\max}}{\sqrt{k_i \cdot m_s}}$ is calculated (where k_i is the internal elasticity of the component), at which the component can be brought down on the PCB. In our model, we have abstracted from the internal forces on the component, but we can still think about safety of the collision by assuming that there is a maximum impact velocity. This is reflected in the following predicate, that describes the condition under which a component remains undamaged:

$$S \equiv (x_1 = x_5 \Rightarrow v_1 - v_5 \geq -v_{\max}).$$

Following the outline for safety analysis introduced in (Cuijpers, 2004), the analysis in the remainder of this paper is aimed at finding parameter values for K, F_{sat}, t_{detect} and v_{detect} such that the following condition holds:

$$\partial_{\neg S}\,(\text{System}) \leftrightarroweq \text{System}.$$

Intuitively, one may say that a system is safe if it never goes into a state in which S is false. Since a change in the value of variables is always visible on the transitions, we may also say that no transitions may occur on which S is false. If (and only if) no transitions can occur on which S is false, then the process System is equivalent[2] to the $\neg S$-encapsulation of System, i.e. the process System with all $\neg S$-transitions removed.

In this paper, we only study damage that results directly from impact. Damage can also occur, for example, after an inelastic impact when $F_{\mathrm{sat}} > F_{\max}$. The study of this and other kinds of damage is left as a topic for future research.

6. ANALYSIS

Due to space considerations, we can only present the outline of our analysis of the pick-and-place module here. This outline is intended to give an overview of the different techniques that were used. For the details of the analysis we refer to (Cuijpers, 2004).

Recall that we set out to investigate under which constraints the condition $\partial_{\neg S}\,(\text{System}) \leftrightarroweq \text{System}$ holds. In this condition, System is formed by a parallel composition of various components. Since the encapsulation operator, in general, does not distribute over parallel compositions, a logical first step in our analysis was to obtain an equivalent representation of System without parallel compositions. This was done using a technique similar to *process algebraic linearization* (Usenko, 2002; van de Brand *et al.*, 2006)[3]. From this, we obtained an equivalent process description of the form

$$\text{System} \leftrightarroweq \begin{bmatrix} x_5^- = v_5^- = 0 \\ margin^- \end{bmatrix} \gg (\text{Seek} \blacktriangleright \text{Detect} \\ \rhd\ impact \odot \text{Bounce})\,,$$

in which Seek, Detect and Bounce are linear subprocesses that each describe the complete systems behavior in one of the three stages of control. The Seek process describes the systems behavior as the sled is brought down, the Detect process describes the systems behavior between collision and the detection of collision. The *impact* action describes the detection of the collision, and the Bounce process describes the systems behavior after detection, when a constant force is applied. Indeed, as the name suggests, the system sometimes performs bouncing behavior in this phase.

As parallel compositions are removed from the system description, elaborate flow clauses arise which describe the physical behavior of the system

[2] As $\leftrightarroweq$ here means *initially stateless bisimulation* in the sense of (Mousavi *et al.*, 2005). This equivalence is a straightforward variant of bisimulation for transition systems in which data plays a role, but in (Cuijpers and Reniers, 2005) it was shown to have congruence problems with respect to parallel composition.

[3] As $\leftrightarroweq$ is not a congruence for parallel composition, we used the stronger notion of *robust bisimulation* (Cuijpers and Reniers, 2005) (also called *stateless bisimulation* in (Mousavi *et al.*, 2005)) for this linearization. The use of two different equivalences indicates that the analysis takes place on two different levels of abstraction. Robust bisimulation is used to reason on an architectural level about the composition of components, while initially stateless bisimulation is used to reason about the behavioral aspects of the system as a whole.

using a set of differential and algebraic equations. These equations were solved using Mathematica, in order to find conditions under which each of the subprocesses, Seek, Detect and Bounce, are safe.

For the subprocesses Seek it was easy to show that the initial conditions $v_{\mathrm{max}} - v_{\mathrm{seek}} \geq v_{\mathrm{detect}}$ and $v_{\mathrm{detect}} \leq \frac{m_{\mathrm{p}}}{m_{\mathrm{s}}} v_{\mathrm{seek}} \leq v_{\mathrm{seek}}$ are sufficient to guarantee safety up to the point of collision.

Showing safety of Detect and Bounce turned out to be feasible only when we assume that collisions between sled and PCB are completely inelastic. Worse, if completely elastic collisions are considered possible, realistic parameter values can be found for which damage might occur at the second impact, i.e. after the first bounce! This is why we aimed at finding conditions under which no second impact could occur.

After changing the Collision process slightly, removing the options for partly elastic collision as described by the fourth and sixth alternative in the definition of Collision in table 2 , we repeated the linearization and found, again, a process of similar form.

But even the assumption of inelastic collisions turned out to be insufficient to guarantee that the sled and PCB stick together after the first impact. The Bounce process still admits bouncing behavior, because after the collision both masses move on together and a counter force builds up that eventually launches the sled of the PCB again. As it turned out, the force with which the sled is pressed onto the PCB should also be large enough to prevent such disconnection. We found a lower bound for this force

$$F_{\mathrm{sat}} \leq \frac{k_{\mathrm{p}} m_{\mathrm{s}}^2 \ v_{\mathrm{max}}}{(m_{\mathrm{s}}+m_{\mathrm{p}})\left(\theta m_{\mathrm{s}}+\omega\sqrt{m_{\mathrm{p}}(m_{\mathrm{p}}+2m_{\mathrm{s}})}\right)},$$

where $\omega = \frac{\sqrt{4k_{\mathrm{p}}(m_{\mathrm{s}}+m_{\mathrm{p}})-b_{\mathrm{p}}^2}}{2(m_{\mathrm{s}}+m_{\mathrm{p}})}$ and $\theta = \frac{b_{\mathrm{p}}}{2(m_{\mathrm{s}}+m_{\mathrm{p}})}$, which may be used as a requirement on the Controller process to guarantee that the sled is pressed to the PCB with sufficient strength, i.e. to guarantee that the Bounce subprocess is safe. Simulations have pointed out that this lower bound can be strengthened further, in some cases up to 60%, by taking the influence of damping into account. However, without insight in how sensitive the lower bound is for changes in the parameters, we must for the time being use the formula given above. Paradoxically, this means that the force that is suggested by Philips CFT in (Mateboer, 1999) should be *increased* by a factor 2 to ensure that the component mounting device is safe. As we have no insight in the maximum static load on the components, we can not determine whether this recommendation is reasonable or not.

Finally, the detection of the impact should be fast enough to guarantee that the pressing phase of the controller is activated before the disconnection of sled and PCB can no longer be prevented. We derived that we need

$$t_{\mathrm{detect}} \leq \frac{\frac{1}{2}\pi - \arctan(\frac{k_{\mathrm{p}}-b_{\mathrm{p}}\theta}{b_{\mathrm{p}}\omega})}{\omega},$$

as a requirement on the Controller to guarantee that the Detect process is safe.

As a last step in our analysis, we need to show that the combination of safe initial conditions of the Seek, Detect and Bounce subprocesses forms a safe initial condition for the System process. For this, we used the special distribution laws for encapsulation and reinitialization given in (Cuijpers, 2004), that allow reasoning over $\Leftrightarrow$ with respect to initial conditions.

7. CONCLUDING REMARKS

In this paper, we have build a hybrid model of a pick-and-place module, using bondgraph components described in the hybrid process algebra HyPA. Furthermore, we have analyzed a control strategy aimed at bringing a component to a PCB as quickly as possible, and press it onto the PCB with sufficient force to make it stick, but without breaking it.

We have shown how HyPA allows us to combine process algebraic analysis and system theoretic analysis, in the sense that these two kinds of analysis have been executed as separate operations on a common hybrid model. Regarding the use of process algebraic methods, we can conclude that the division of the system as a whole into manageable subprocesses certainly helped in the analysis of the pick-and-place module. Another strong point, is that we were able to describe impact detection as a separate sequential process. This allowed a separate formal treatment of the different phases of the process, which was useful because timing issues that played a role in the Detect process would otherwise have interfered with the search for a constraining force in the Bounce process. Also the collision dynamics were modeled as a separate component, which allowed us to easily change the model when it became apparent that we needed to assume inelastic collisions. This would not have been possible if we had not given semantics to the bondgraph paradigm using hybrid constitutive processes.

A weak point in the analysis, is that we needed to solve many of the differential equations that occur in the subprocesses. This made it hard to generalize our results to an analysis method, because many differential equations that occur in practice cannot be solved analytically. On the other hand, it was exactly the use of hybrid constitutive processes that allowed us to describe the physical behavior of the module using only simple differential equations in the first place. If we had not used the hybrid constitutive processes to model the module at exactly the right level of abstraction, the differential equations would certainly have been too difficult to solve. In particular, the hybrid approach allowed us to use basic conservation laws only to describe the impact behavior, instead of resorting to complex differential equations.

In the analysis discussed in this paper, HyPA was used to transfer analysis results from system theory to process algebra and back. The core of the safety-analysis was a proof that the system satisfies a certain process algebraic condition. Proving that this condition holds was done along process algebraic lines of reasoning, but each iteration required a system theoretic proof about the differential equations involved. In (Cuijpers, 2004), this method was already outlined, but as a topic for future research we propose to search for more and better combined proof methods of this kind, with a focus on hybrid topics like safety, stability, liveness and freedom of deadlocks.

ACKNOWLEDGEMENTS

Many thanks go to Paul van den Bosch, Jan Friso Groote and Ka Lok Man, for proof reading a preliminary version of this paper.

REFERENCES

Alur, R. and D.L. Dill (1994). A theory of timed automata. *Theoretical Computer Science* **126**(2), 183–235.

Baeten, J.C.M. and W.P. Weijland (1990). *Process Algebra*. Vol. 18 of *Cambridge Trancts in Theoretical Computer Science*. Cambridge University Press. Cambridge.

Bergstra, J.A. and C.A. Middelburg (2005). Process algebra for hybrid systems. *Theoretical Computer Science* **335**(2-3), 215–280.

Brinksma, E. (1985). A tutorial on LOTOS. In: *Proc. Protocol Specification, Testing and Verification V* (Michel Diaz, Ed.). Amsterdam, Netherlands. pp. 171–194.

Cuijpers, P.J.L. (2004). Hybrid Process Algebra. PhD thesis. Technische Universiteit Eindhoven (TU/e). Eindhoven, Netherlands.

Cuijpers, P.J.L. and M.A. Reniers (2005). Hybrid process algebra. *Journal of Logic and Algebraic Programming* **62**(2), 191–245.

Cuijpers, P.J.L., J.F. Broenink and P.J. Mosterman (2004). Constitutive hybrid processes. In: *Conference on Conceptual Modeling and Simulation*. Genua, Italy.

Dorf, R.C. and R.H. Bishop (1995). *Modern Control Systems*. Series in Electrical and Computer Engineering: Control Engineering. Addison-Wesley.

Febbraro, A. Di, Giua, A. and Menga, G., Eds. (2001). *Special Issue on Hybrid Petri Nets*. Vol. 11 of *Discrete Event Dynamic Systems*.

Heemels, W.P.M.H., B. De Schutter and A. Bemporad (2001). On the equivalence of classes of hybrid dynamical models. In: *Proc. Conference on Decision and Control*. Orlando, Florida. pp. 364–369.

Henzinger, T.A. (1996). The theory of hybrid automata. In: *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science (LICS 1996)*. IEEE Computer Society Press. pp. 278–292.

Karnopp, D.C., D.L. Margolis and R.C. Rosenberg (1990). *System Dynamics: A Unified Approach*. John Wiley and Sons, Inc.

Man, K., M.A. Reniers and P.J.L. Cuijpers (2005). Case studies in the hybrid process algebra hypa. *International Journal of Software Engineering and Knowledge Engineering* **15**(2), 299–305.

Mateboer, A.J. (1999). Eindverslag phi-z. Technical Report CTB595-99-3044. Philips CFT. Eindhoven, Netherlands.

Mosterman, P., G. Biswas and O. Otter (1998). Simulation of discontinuities in physical system models based on conservation principles. In: *Proceedings of the 1998 Summer Computer Simulation Conference*. Reno, Nevada. pp. 320–325.

Mousavi, M.R., M.A. Reniers and J.F. Groote (2005). Notions of bisimulation and congruence formats for SOS with data. *Information and Computation (I&C)* **200**(1), 104–147.

Rounds, W.C. and H. Song (2003). The ϕ-calculus: A language for distributed control of reconfigurable embedded systems. In: *Hybrid Systems: Computation and Control, 6th International Workshop, HSCC 2003* (F. Wiedijk, O. Maler and A. Pnueli, Eds.). Vol. 2623 of *Lecture Notes in Computer Science*. Springer-Verlag. pp. 435–449.

Usenko, Y.S. (2002). Linearization in μCRL. PhD thesis. Technische Universiteit Eindhoven (TU/e).

van Beek, D.A., K.L. Man, M.A. Reniers, J.E. Rooda and R.R.H. Schiffelers (2006). Syntax and consistent equation semantics of hybrid chi. *Journal of Logic and Algebraic Programming* **68**(1-2), 129–210.

van de Brand, P., M.A. Reniers and P.J.L. Cuijpers (2006). Linearization of hybrid processes. *Journal of Logic and Algebraic Programming* **68**(1-2), 54–104.

van der Schaft, A.J. and J.M. Schumacher (2000). *An Introduction to Hybrid Dynamical Systems*. Vol. 251 of *Lecture Notes in Control and Information Sciences*. Springer-Verlag. London.