Modeling and Analysis of Semiconductor Manufacturing Systems with Degraded Behavior

Using Petri Nets and Siphons

MuDer Jeng
Department of Electrical Engineering
National Taiwan Ocean University
Keelung 202, Taiwan, ROC

Xiaolan Xie
LGIPM and INRIA/MACSI Team
ENIM-Ile du Saulcy
F-57045 Metz Cedex 1, France

Abstract
Degraded behavior, such as reworks, failures, and maintenance, of a semiconductor manufacturing system (SMS) is not negligible in practice. When modeled by Petri nets, degraded behavior may be represented as initially-unmarked elementary circuits, interpreted as local processing cycles. Most existing “well-behaved” net classes for manufacturing have problems of describing such cycles and thus may have difficulties in modeling SMSs. In this paper, we extend the class of nets in [6] into the class of RCN* merged nets that model SMSs with such cycles. To model an SMS, we first describe the behavior of each resource type using a state-machine module, called RCN. Any RCN can be constructed as a connection of acyclic sub-nets called blocks, where one of them denotes the normal behavior of the resource type and the others denote its degraded behavior. Next, an RCN* merged net for the entire system is built by fusing all modules, conforming to three constraints, along their common transition sub-nets, which represent their synchronization. In the analysis of RCN* merged nets, we prove that their liveness and reversibility depend on the absence of unmarked siphons, which are structural objects that mixed integer programming can check rapidly. Examples are given to illustrate the proposed approach.

1. Introduction
Semiconductor manufacturing systems (SMSs) belong to a class of discrete manufacturing systems that contain complicated production procedures and a large number of shared resources. In particular, an SMS often reveals degraded behavior, which is not negligible in practice. For example, if in the photolithography stage a wafer is not properly coated with photo-resists, it is necessary to remove the resistors and re-do the coating. On the other hand, when a wafer is waiting to be processed in some machine, the machine may not be available due to some unexpected fault or expected maintenance operation. These are three examples of degraded behavior -- reworks, failures, and maintenance. The purpose of this paper is to present a new class of Petri nets for the modeling and analysis of SMSs with such behavior

Petri nets [16] have become a powerful tool for manufacturing after three decades of research and development. Recently, some researchers have applied them to SMSs. Kim and Desrochers [9] investigated the automatic generation and performance evaluation of a net model for an SMS based on the technology process flows and fabrication facilities. In [1, 13], Petri nets and heuristic search were adopted for modeling and scheduling an SMS. In [5, 11, 12], the modeling and analysis of several types of equipment in SMSs was performed using Petri nets. Cheng et al. [2] generated and analyzed a net model for a semiconductor manufacturing execution system from IDEF0 and state diagrams. In the above-mentioned approaches, degraded behavior of SMSs was not explicitly considered at the net structure level. Yoon and Lee [14] exploited a very simple failure-repair net model for lot dispatching in a wafer fabrication facility. Degraded behavior is explicitly considered in [16], where a class of modularly composed Petri nets was proposed for modeling, qualitative analysis, and performance evaluation of a real-world IC fabrication system under the assumption that there exists exactly one global buffer. Combined with the uniformization technique, the class of nets was adopted for analytically evaluating SMSs in [8].

Since an SMS is very complex, analyzing its model for qualitative properties is a computation-intensive job. As a result, our focus in this paper is to explore a class of “well-behaved” nets for SMSs that can explicitly describe degraded behavior at the net structure level and whose properties can be checked within reasonable time. Previous “well-behaved” net classes may have difficulties in applications to SMSs because some are inflexible to represent shared resources (e.g. [10]), some take significant computation time in qualitative analysis (e.g. [15]), and others have problems to denote degraded behavior (e.g. [4, 6]). In terms of net structure, degraded behavior may be represented as initially-unmarked elementary circuits, interpreted as local processing cycles. Nevertheless, this does not mean that the last category of net classes mentioned above [4, 6] cannot describe such cycles. Through the transformation or reduction technique, they can denote local processing cycles that involve only one resource type as in [14]. In this paper, we present a new net class called RCN* merged nets that can describe local processing cycles that involve more than one resource type. Comparing to the nets in [8], RCN* merged nets do not require the assumption that there exists exactly one global buffer.

To model an SMS, we first describe the behavior of
each resource type (e.g., wafer, machine, buffer, AGV, etc.) using a state-machine module, called RCN. The concept of resource used here encompasses both the resources for fabricating the parts, e.g., wafers, and the parts themselves. Any RCN can be constructed as a connection of acyclic sub-nets called blocks, where one of them denotes the normal behavior of the resource type and the others denote its degraded behavior. The connection of any two blocks is accomplished by using degrading and restoration arcs. Next, an RCN merged net for the entire system is built by fusing all modules, conforming to three constraints, along their common transition sub-nets, which represent their synchronization. In fact, we have generalized the nets in [6] into RCN merged nets by relaxing one of the original constraints.

We show that liveness and reversibility of RCN merged nets depend on the absence of unmarked siphons, which are structural objects much easier to check than the state-space based conditions. It is worth mentioning that a mathematical programming approach was proposed in [3] to avoid explicit enumeration of siphons. It allows one to check large Petri net models using powerful mathematical programming software packages. It is reported that this approach can check large nets of about 200 places and 200 transitions in a small amount of time. A net with such size can model significantly complex SMSs.

The remaining paper is organized as follows: Section 2 defines RCN merged nets. Section 3 discusses their qualitative properties. Concluding remarks and future research are presented in Section 4. The appendix briefly summarizes the Petri net theory.

2. RCN Merged Nets

As described above, the class of RCN merged nets is constructed in a resource oriented and modular manner as the class of nets in [6]. The main construction steps include the modeling of the behavior of each resource type using Petri nets, and the integration of resource net modules by taking into account interactions among resource types. Each resource type is modeled as an RCN, constructed as a connection of blocks, and the integration is realized through merging of the RCNs along their common transition sub-nets, where the common elements of any two RCNs are denoted as having the same labels. Here the concept of resources includes both the resources for fabricating parts, e.g., wafers, and the parts themselves.

The following definition of an RCN was given in [6, 7].

Definition 1: An RCN $G_\alpha = (P_\alpha, T_\alpha, F_\alpha, M_{\alpha0})$ is a strongly connected state machine where there exists exactly one place $p_{\alpha0} \in P$, called resource place, such that $M_{\alpha0}(p_{\alpha0}) \neq 0$. The remaining places are called operation places.

Place $p_{\alpha0}$ denotes the availability of the resource type $R_\alpha$ that $G_\alpha$ models. In most manufacturing systems, there exists a state such that all resources are available and there is no work-in-process. We choose such a state as the initial state. It is then reasonable that only the resource places are initially marked. The portion of $G_\alpha$ excluding $p_{\alpha0}$ represents operations that require $R_\alpha$.

Remark 1: An operation place with its input transition(s) and output transition(s) models some operation $op_x$ being executed, the start of $op_x$, and the end of $op_x$, respectively, as shown in Fig. 1. Each such transition(s)-place transition(s) structure represents an operation with a specific resource requirement. We can build an RCN by exploiting such net structures and by considering the precedence relationship among the operations, i.e., the input transitions for some operation are the output transitions for its previous operations (see Example 1 later).

Fig. 1. Modeling of an operation.

The above definition of an RCN does not explicitly specify how to represent the degraded behavior of a resource type. Here we define degraded behavior as having local processing cycles:

Definition 2: A local processing cycle is a circuit that contains operation places only, while a global processing cycle is thus a circuit that contains at least one resource place.

The above definition applies to individual RCNs as well as integrated nets, as presented later. Note that the nets in [6] prohibit the modeling of local processing cycles at both the module and the integrated net levels due to one of their restrictions that the removal of all resource places from an integrated net results in an acyclic net. This prevents any RCN of the nets in [6] from containing local processing cycles.

To model the normal and degraded behavior of a resource type, any RCN can be constructed as a connection of blocks as shown in Fig. 2, where the blocks, which are defined below, are acyclic sub-nets and the double circle denotes the resource place $p_r$. Block 1 denotes the normal behavior of a semiconductor resource type while block $x$ represents its degraded behavior due to, for example, $(x-/)$ times of reworks, failures, or maintenance.

Definition 3: The blocks of an RCN $G$ are defined as follows:

Step 1: Let $T_1^\alpha = p_\alpha$ and $T_0^\alpha = \bullet p_\alpha$ be respectively called the sets of input and output transitions of
block 1, denoted as \( B_1^i(G) \). Then \( B_1^i(G) = (P_{B_1}^i, T_{B_1}^i, F_{B_1}^i, M_{B_1}^i) \) is a net defined as the maximal circuit-free union of elementary paths from \( T_{B_1}^i \) to \( T_{B_1}^i \) (of course, excluding \( p_i \)).

Step 2: Repeat the following step for \( x = 2, \ldots \) until no new block is found.

Step 3: Let \( T_1^g = \bigcup_{p \in P_{B_i}^g \cup T_{B_i}} \bigcup_{p \in P_{B_i}^{g+1}} T_{B_i} \) and \( T_0^g = \bigcup_{p \in P_{B_1}^1 \cup T_{B_1}} \bigcup_{p \in P_{B_1}^{1+1}} T_{B_1} \) be respectively called the sets of input and output transitions of block \( x \), denoted as \( B_x^i(G) \). Then \( B_x^i(G) = (P_{B_x}^i, T_{B_x}^i, F_{B_x}^i, M_{B_x}^i) \) is a net defined as the maximal circuit-free union of elementary paths from \( T_{B_x}^i \) to \( T_{B_x}^i \).

In the following, places in \( \bullet T_{B_x}^i \cap P_{B_{x-1}} \) will be called interface places (e.g., \( p_1 \) in Fig. 2) of block \((x-1)\) with \( x > 1 \). Arcs \( (p, t) \) from block \( B_{x-1}^i(G) \) to \( B_x^i(G) \) will be called degrading arcs and arcs \( (p, t) \) from block \( B_x^i(G) \) to \( B_{x-1}^i(G) \) will be called restoration arcs. Of course, any degrading arc connects an interface place to an input transition of the next block while a restoration arc connects an output transition to the previous block.

**Example 1:** Fig. 3, where the black dot denotes the initial state, shows a typical photolithography process for some wafer type \( W \) using online steppers. Wafers are initially coated with photo-resistor circuit patterns. Then, the photo resistors are exposed and developed. Finally, they are inspected to see if their dimensions are correct or not. If not, the resistors on the wafers will be removed and the wafers are re-coated with new resistors. During the process, wafers are usually stored in buffers between two consecutive processing steps. Inline steppers are modern wafer-processing machines that can successively handle coating, exposure, and development in three chambers, respectively. In addition, they can perform resistors removal and re-coating in the coating chamber.

The process flow for \( W \) has degraded behavior for re-coating. Based on Fig. 3, we can model \( W \) as the RCN \( G_W \) in Fig. 4 (w denotes the initial marking of the resource place), where a dashed block models an operation, and blocks 1 and 2 (shaded) denote the normal process and the rework process respectively. In Fig. 4, we are aware of a local processing cycle \( p_3t_4p_4t_5t_6p_6t_9p_9t_10t_11p_3 \) due to the degraded behavior for re-coating. [17] describes a system for such a process. The system contains a set of inline steppers \( L \), two types of buffers \( B_1, B_2 \), and a set of inspection machines \( E \). Due to space limitation, the detailed modeling of the system using the proposed approach is left out.

![Fig. 2](image1)

**Fig. 2.** The blocks within an RCN.

In Fig. 2, the blocks are connected as follows: Block 1 connects to Block 2 and Block 2 connects to Block 3. The blocks are denoted as \( B_1, B_2, B_3 \). The places in the blocks are denoted as \( p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_10, p_11 \). The transitions in the blocks are denoted as \( t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{11} \). The arcs in the blocks are denoted as \( (p_i, t_j) \). The figure illustrates the flow of the process from the coating chamber to the inspection chamber.

![Fig. 3](image2)

**Fig. 3.** A typical photolithography process for some wafer type \( W \).

The following lemma shows an important property of blocks.

**Lemma 1:** Each block \( B_i(G) \) in an RCN \( G \) is acyclic. Further, there exist a path in \( B_i(G) \) from any given input transition to an output transition and a path from an input transition to any given output transition.

To build a system net, RCNs are merged along their common elements. Since each RCN is constructed according to Remark 1, it is clear that the common elements of any two RCNs form a transition sub-net, as given below [6, 7]:

**Definition 4:** A transition sub-net \( G_p = (P_p, T_p, F_p, M_p) \) of a Petri net \( G \) is a sub-net of \( G \) such that input transitions and output transitions of any place \( p \in P_p \) (\( P_p \) can be empty) are transitions in \( T_p \). In other words, the places of a transition sub-net have no arc connection outside the sub-net.

![Fig. 4](image3)

**Fig. 4.** RCN \( G_W \) for the wafer type \( W \).
Definition 5: Given a set of n RCNs \( \{G_s \mid G_s = (P_s, T_s, F_s, M_{0s}), s = 1, \ldots, n \} \) built following Remark 1, an RCN* merged net \( G = (P, T, F, M_0) \) is their union, i.e., \( P = P_1 \cup P_2 \cup \ldots \cup P_n, T = T_1 \cup T_2 \cup \ldots \cup T_n, F = F_1 \cup F_2 \cup \ldots \cup F_n \) and \( M_0(p) = M_0(p) \) if \( p \in P_1 \), satisfying the following three restrictions.

Restriction 1: At each common transition, there exists at most one input place that is an operation place.

Restriction 2: Common transition sub-nets should not include resource places.

Restriction 3: \( (A) \) In the integrated model \( G \), for each place \( p \) with degrading outgoing arcs, there exist a non-degrading outgoing arc \( (p, t_1) \) such that \( \bullet t_1 \subseteq \bullet t_2 \) for any degrading outgoing arc \( (p, t_2) \). \( (B) \) The net \( G' \) derived from \( G \) by removing resource places has source and sink transitions, also called global input and output transitions, such that there exists a path in \( G' \) from an input transition to any node of \( G' \). \( (C) \) The net \( G# \) derived from \( G \) by removing the resource places and degrading arcs is an acyclic graph.

Restrictions 1 and 2 also hold for the nets in [6]. We briefly repeat the explanations given in [6] for completeness.

Restriction 1 excludes the modeling of the synchronization of parallel processes or the assembly of several components. Thus, it is restrictive and its relaxation is an issue of future research. Restriction 2 is natural for our resource-oriented approach since each resource type is modeled as exactly one RCN. That is, each resource place appears in exactly one RCN.

Restriction 3 here is more general than that for the nets in [6] because the integrated model can contain circuits that do not include the resources places as long as the “AC (asymmetric choice)” condition (i.e., \( \bullet t_1 \subseteq \bullet t_2 \) above) is satisfied. In other words, all such circuits, i.e., local processing cycles, include such transitions \( t_2 \).

Restriction 3 is reasonable as explained below. In most manufacturing systems, local processing cycles are due to operation constraints (e.g., reworks, failures, and maintenance) rather than resource constraints (e.g., the availability of resources). Thus, the decision of firing \( t_1 \) (generating a global processing cycle) or \( t_2 \) (generating a local processing cycle) at an interface place \( p \) is usually made in an “FC (free-choice)” manner, i.e., \( [p] = \bullet t_1 = \bullet t_2 \).

In general, such decisions are based on information independent of the availability of resources such as product data, product quality or degrading state of equipments. In other words, no resources are involved in such a decision. For the same set of resources used at \( t_1 \) and \( t_2 \), this decision can be reasonably generalized to “extended FC”, i.e., \( \bullet t_1 = \bullet t_2 \). For example, in a semiconductor manufacturing system, a wafer may be placed in the same general buffer when either it is processed for the next operation, denoted by \( t_1 \), or reworked for a previous operation, denoted by \( t_2 \). In this condition, a local processing cycle is still, in effect, generated by operation constraints at the operation place \( p \). We further extend the “extended FC” condition to the “AC” condition, leading to Restriction 3A. In the latter condition, a local processing cycle is more constrained to be created since a rework, failure, or maintenance operation may involve a larger set of resources than a normal operation does. For example, an extra robot is required for failure repair or maintenance.

As for Restrictions 3B and 3C, we consider that local processing cycles are usually the degrading behaviors of single resource (i.e., RCN) since they may be interpreted as reworks, failures or maintenance. Consequently, Restriction 3C excludes circuits that result from the normal operational behaviors without degrading arcs. For example, the net in Fig. 5 is not an RCN* merged net due to the circuit \( t_1 p_1 t_2 p_2 t_1 \) that contains all operation places, which are initially unmarked, where dotted blocks are RCNs, and \( t_1 \) and \( t_2 \) are the common transitions. Restriction 3B restricts processing cycles involving more than one RCN to be controllable via global input transitions. Local cycles such as the one of Fig. 6 create confusion between normal behaviors and degrading behaviors. In Fig. 6, \( t_1 \) belongs to the normal behavior and \( t_2 \) the degrading behavior in the left RCN while the reverse happens in the right RCN. Such confusion leads to a deadlock situation as proved in Lemma 2, given later.

By construction, an RCN* merged net \( G \) is state machine decomposable. Each RCN is a state machine component. The number of tokens in any state machine component remains constant whatever transition firings. Hence,

**Property 1:** \( G \) is conservative and structurally bounded. The following lemmata are due to Restriction 3:

**Lemma 2:** \( G \) contains an empty siphon at the initial marking and hence is not live if Restriction 3B does not hold.

**Lemma 3:** Under Restrictions 1 and 3B, any siphon in \( G \) contains at least one resource place.

### 3. Qualitative Properties

This section discusses two important qualitative properties, reversibility and liveness, of an RCN* merged net \( G \), which satisfies Restrictions 1-3.
Theorem 1: Under Restrictions 1-3, G is reversible iff no siphons in G can become unmarked.

Notice that the condition of Theorem 1 can be checked using the following property obtained in [3]:

Property 2: A siphon S can never become empty if it contains a marked trap or F(S) > 0 with F(S) = \min{\sum_{M \in \rho_S} \{M = M_0 + CY, M \geq 0, Y \geq 0\}}.

Beside the above property, we can exploit the mathematical programming approach proposed in [3] to avoid explicit enumeration of siphons. Experimental results reported in [3] show that using powerful mathematical programming software packages, the approach is able to check large nets of about 200 places and 200 transitions in a small amount of time.

Under the reversibility, the liveness of the RCN* merged net G is reduced to its potential liveness, a property much easier to check.

Theorem 2: Under Restrictions 1-3, G is live and reversible iff no siphon in G can become unmarked and every transition can fire at least once, i.e., every transition is potentially firable.

The following results can be used to check the potential firability of the transitions:

Restriction 4: At any common transition, there is at most one output place that is an operation place.

Theorem 3: Under Restrictions 1-4, any transition in G is potentially firable.

The proof of this theorem is similar to that of Theorem 3 in [6], it is omitted and only a sketch of proof is given. The proof is based on the existence of an elementary path from a global input transition to any transition t composed of only operation places. Restrictions 1 and 4 ensure that transitions on this path do not have other operation input/output places. Firability of these transitions can then be easily proved.

Consider the case where Restriction 4 is not satisfied. It is more difficult to check the potential firability in this case since any transition not conforming to Restriction 4 creates parallel processes, i.e., more than one output operation place. The potential firability is not always true as shown in Fig. 7 where the RCN* merged net is obtained by composing two RCNs sharing t1, t2, and t3.

Fig. 6. A net with an inter-RCN cycle.

Theorem 1: Under Restrictions 1-3, G is reversible iff no siphons in G can become unmarked.

Property 2: A siphon S can never become empty if it contains a marked trap or F(S) > 0 with F(S) = \min{\sum_{M \in \rho_S} \{M = M_0 + CY, M \geq 0, Y \geq 0\}}.

Beside the above property, we can exploit the mathematical programming approach proposed in [3] to avoid explicit enumeration of siphons. Experimental results reported in [3] show that using powerful mathematical programming software packages, the approach is able to check large nets of about 200 places and 200 transitions in a small amount of time.

Under the reversibility, the liveness of the RCN* merged net G is reduced to its potential liveness, a property much easier to check.

Theorem 2: Under Restrictions 1-3, G is live and reversible iff no siphon in G can become unmarked and every transition can fire at least once, i.e., every transition is potentially firable.

The following results can be used to check the potential firability of the transitions:

Restriction 4: At any common transition, there is at most one output place that is an operation place.

Theorem 3: Under Restrictions 1-4, any transition in G is potentially firable.

The proof of this theorem is similar to that of Theorem 3 in [6], it is omitted and only a sketch of proof is given. The proof is based on the existence of an elementary path from a global input transition to any transition t composed of only operation places. Restrictions 1 and 4 ensure that transitions on this path do not have other operation input/output places. Firability of these transitions can then be easily proved.

Consider the case where Restriction 4 is not satisfied. It is more difficult to check the potential firability in this case since any transition not conforming to Restriction 4 creates parallel processes, i.e., more than one output operation place. The potential firability is not always true as shown in Fig. 7 where the RCN* merged net is obtained by composing two RCNs sharing t1, t2, and t3.

Fig. 6. A net with an inter-RCN cycle.

Theorem 1: Under Restrictions 1-3, G is reversible iff no siphons in G can become unmarked.

Property 2: A siphon S can never become empty if it contains a marked trap or F(S) > 0 with F(S) = \min{\sum_{M \in \rho_S} \{M = M_0 + CY, M \geq 0, Y \geq 0\}}.

Beside the above property, we can exploit the mathematical programming approach proposed in [3] to avoid explicit enumeration of siphons. Experimental results reported in [3] show that using powerful mathematical programming software packages, the approach is able to check large nets of about 200 places and 200 transitions in a small amount of time.

Under the reversibility, the liveness of the RCN* merged net G is reduced to its potential liveness, a property much easier to check.

Theorem 2: Under Restrictions 1-3, G is live and reversible iff no siphon in G can become unmarked and every transition can fire at least once, i.e., every transition is potentially firable.

The following results can be used to check the potential firability of the transitions:

Restriction 4: At any common transition, there is at most one output place that is an operation place.

Theorem 3: Under Restrictions 1-4, any transition in G is potentially firable.

The proof of this theorem is similar to that of Theorem 3 in [6], it is omitted and only a sketch of proof is given. The proof is based on the existence of an elementary path from a global input transition to any transition t composed of only operation places. Restrictions 1 and 4 ensure that transitions on this path do not have other operation input/output places. Firability of these transitions can then be easily proved.

Consider the case where Restriction 4 is not satisfied. It is more difficult to check the potential firability in this case since any transition not conforming to Restriction 4 creates parallel processes, i.e., more than one output operation place. The potential firability is not always true as shown in Fig. 7 where the RCN* merged net is obtained by composing two RCNs sharing t1, t2, and t3.

Fig. 6. A net with an inter-RCN cycle.

Theorem 1: Under Restrictions 1-3, G is reversible iff no siphons in G can become unmarked.

Property 2: A siphon S can never become empty if it contains a marked trap or F(S) > 0 with F(S) = \min{\sum_{M \in \rho_S} \{M = M_0 + CY, M \geq 0, Y \geq 0\}}.

Beside the above property, we can exploit the mathematical programming approach proposed in [3] to avoid explicit enumeration of siphons. Experimental results reported in [3] show that using powerful mathematical programming software packages, the approach is able to check large nets of about 200 places and 200 transitions in a small amount of time.

Under the reversibility, the liveness of the RCN* merged net G is reduced to its potential liveness, a property much easier to check.

Theorem 2: Under Restrictions 1-3, G is live and reversible iff no siphon in G can become unmarked and every transition can fire at least once, i.e., every transition is potentially firable.

The following results can be used to check the potential firability of the transitions:

Restriction 4: At any common transition, there is at most one output place that is an operation place.

Theorem 3: Under Restrictions 1-4, any transition in G is potentially firable.

The proof of this theorem is similar to that of Theorem 3 in [6], it is omitted and only a sketch of proof is given. The proof is based on the existence of an elementary path from a global input transition to any transition t composed of only operation places. Restrictions 1 and 4 ensure that transitions on this path do not have other operation input/output places. Firability of these transitions can then be easily proved.

Consider the case where Restriction 4 is not satisfied. It is more difficult to check the potential firability in this case since any transition not conforming to Restriction 4 creates parallel processes, i.e., more than one output operation place. The potential firability is not always true as shown in Fig. 7 where the RCN* merged net is obtained by composing two RCNs sharing t1, t2, and t3.

Fig. 6. A net with an inter-RCN cycle.

Theorem 1: Under Restrictions 1-3, G is reversible iff no siphons in G can become unmarked.

Property 2: A siphon S can never become empty if it contains a marked trap or F(S) > 0 with F(S) = \min{\sum_{M \in \rho_S} \{M = M_0 + CY, M \geq 0, Y \geq 0\}}.

Beside the above property, we can exploit the mathematical programming approach proposed in [3] to avoid explicit enumeration of siphons. Experimental results reported in [3] show that using powerful mathematical programming software packages, the approach is able to check large nets of about 200 places and 200 transitions in a small amount of time.

Under the reversibility, the liveness of the RCN* merged net G is reduced to its potential liveness, a property much easier to check.

Theorem 2: Under Restrictions 1-3, G is live and reversible iff no siphon in G can become unmarked and every transition can fire at least once, i.e., every transition is potentially firable.

The following results can be used to check the potential firability of the transitions:

Restriction 4: At any common transition, there is at most one output place that is an operation place.

Theorem 3: Under Restrictions 1-4, any transition in G is potentially firable.

The proof of this theorem is similar to that of Theorem 3 in [6], it is omitted and only a sketch of proof is given. The proof is based on the existence of an elementary path from a global input transition to any transition t composed of only operation places. Restrictions 1 and 4 ensure that transitions on this path do not have other operation input/output places. Firability of these transitions can then be easily proved.

Consider the case where Restriction 4 is not satisfied. It is more difficult to check the potential firability in this case since any transition not conforming to Restriction 4 creates parallel processes, i.e., more than one output operation place. The potential firability is not always true as shown in Fig. 7 where the RCN* merged net is obtained by composing two RCNs sharing t1, t2, and t3.
component \( G_1 \) of a net \( g^* \) is a sub-net generated by a sub-set \( T_1 \) of transitions having the following properties: (1) each place in \( G_1 \) has one output transition and at least one input transition; (2) a sub-net generated by \( T_1 \) is the net consisting of transitions in \( T_1 \), all of their input and output places, and their connecting arcs.

**Definition 9:** Let \( G_1 = (P_1, T_1, F_1, M_{10}) \) be an FCF component of the net \( g^* \). \( f(G_1) \) is defined as the sub-net of \( G^* \) generated by transitions in \( T_1 \). Clearly, \( f(G_1) \) can be derived from \( G_1 \) by adding resource places that are input or output places of transitions in \( T_1 \).

As a result of Lemmata 5 and 6, we can exploit the properties of the class of nets in [6] to verify the potential firability of each transition in an RCN* merged network.

**Theorem 4:** A transition \( t \) in \( G \) is potentially firable if and only if there exists an FCF component \( G_1 \) in \( g^* \) that contains \( t \) and that no siphon in \( f(G_1) \) can become unmarked.

### 4. Conclusions

In this paper, we have presented a new class of modularly composed Petri nets called RCN* merged nets that model semiconductor manufacturing systems with degraded behaviors such as reworks, failures, and maintenance. RCN* merged nets are generalized from the class of nets in [6]. Degraded behaviors are denoted as local processing cycles, which are elementary circuits that are initially unmarked. To model degraded behaviors, an RCN module can be built as a connection of blocks, where one block denotes the normal behavior and the others denote the degraded behaviors. The liveness and reversibility of RCN* merged net are proved to depend on the absence of unmarked siphons. Future research is directed to the extension to manufacturing systems with assembly/disassembly operations and degrading behaviors.

### References


