

12-31-2022

## Modeling of quad-station module cluster tools using petri nets

Aung Nay

*New Jersey Institute of Technology*, an553@njit.edu

Follow this and additional works at: <https://digitalcommons.njit.edu/theses>



Part of the [Computer Engineering Commons](#), [Dynamics and Dynamical Systems Commons](#), and the [Electrical and Computer Engineering Commons](#)

---

### Recommended Citation

Nay, Aung, "Modeling of quad-station module cluster tools using petri nets" (2022). *Theses*. 2096.  
<https://digitalcommons.njit.edu/theses/2096>

This Thesis is brought to you for free and open access by the Electronic Theses and Dissertations at Digital Commons @ NJIT. It has been accepted for inclusion in Theses by an authorized administrator of Digital Commons @ NJIT. For more information, please contact [digitalcommons@njit.edu](mailto:digitalcommons@njit.edu).

## **Copyright Warning & Restrictions**

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be “used for any purpose other than private study, scholarship, or research.” If a user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of “fair use” that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

**Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation**

Printing note: If you do not wish to print this page, then select “Pages from: first page # to: last page #” on the print dialog screen

The Van Houten library has removed some of the personal information and all signatures from the approval page and biographical sketches of theses and dissertations in order to protect the identity of NJIT graduates and faculty.

## **ABSTRACT**

### **MODELING OF QUAD-STATION MODULE CLUSTER TOOLS USING PETRI NETS**

**by  
Aung Nay**

The semiconductor industry is highly competitive, and with the recent chip shortage, the throughput of wafers has become more important than ever. One of the tools that the industry has deployed is to use of quad-station modules instead of the traditional single-station modules that allow for higher throughput and better wafer consistency by processing multiple wafers at the same time and distributing work. The industry trend is to use multiple transfer chamber robots to stack the quad-station modules in a series, particularly for etch products. In this work, the quad-station cluster tool wafer movement is modeled by using Petri net as a process-bounded system. The system analysis and simulations are performed by using timed and colored Petri nets. The results are useful to deepen our understanding of the discrete-event dynamics of quad-station module cluster tools and offer the highly needed insight into their efficient and deadlock-free operation.

**MODELING OF QUAD-STATION MODULE CLUSTER TOOLS  
USING PETRI NETS**

**by  
Aung Nay**

**A Thesis  
Submitted to the Faculty of  
New Jersey Institute of Technology  
in Partial Fulfillment of the Requirements for the Degree of  
Master in Computer Engineering**

**Department of Electrical and Computer Engineering**

**December 2022**



**APPROVAL PAGE**

**MODELING OF QUAD-STATION MODULE CLUSTER TOOLS  
USING PETRI NETS**

**Aung Nay**

---

Dr. Mengchu Zhou, Dissertation Advisor Date  
Distinguished Professor of Electrical and Computer Engineering, NJIT

---

Dr. Tao Han, Committee Member Date  
Associate Professor of Electrical and Computer Engineering, NJIT

---

Dr. Xiwang Guo, Committee Member Date  
Associate Professor of Computer and Communication Engineering College, Liaoning  
Petrochemical University, Fushun, 113001, P R. China

## **BIOGRAPHICAL SKETCH**

**Author:** Aung Nay

**Degree:** Master

**Date:** December 2022

**Date of Birth:**

**Place of Birth:**

**Undergraduate and Graduate Education:**

- Master of Science in Computer Engineering,  
New Jersey Institute of Technology, Newark, NJ, 2022
- Bachelor of Arts in Economics,  
The Ohio State University, Columbus, OH, 2007

**Major:** Computer Engineering



To my partner in good and bad, 🤝!

## ACKNOWLEDGMENT

This work would not have been possible without Prof. Mengchu Zhou's inspiration, direction, and guidance. I am very much in debt to Prof. Zhou for his generosity, for the opportunities he has freely shared, and for being a great teacher who hooked me into the subject matter more than any other teacher I have ever had.

I am also very thankful to Dr. Side Zhao for giving me a glimpse into the semiconductor industry with a warm welcome and opening the door to a wonderful world full of good human beings.

I would also like to thank the committee members: Prof. Xiwang Guo and Prof. Tao Han, for being so generous and helping with the process.

I am very grateful to all my friends who worked with me along this learning journey: Megha, Salman, Aftab, Cynthia, Ravi, and Brendan.

I want to thank my mom and dad for giving me opportunities to build foundational knowledge and developing my ability to persevere.

Most importantly, I thank the most important individual in my life, my wife, Aye. She has been a constant source of support, encouragement, and joy through challenges. She inspires me to aspire to be a little better every day. I cannot imagine my life without Aye.

## TABLE OF CONTENTS

Chapter	Page
1 INTRODUCTION.....	1
1.1 Overview .....	1
1.2 Cluster Tool Background .....	2
2 PETRI NET OVERVIEW .....	4
2.1 Petri Net Basics .....	4
2.2 A Marked Petri Net Formal Definition .....	6
2.3 Anti-Place Definition .....	7
3 CLUSTER TOOL OVERVIEW .....	9
3.1 Description of Cluster Tool .....	9
4 PETRI NET MODELS OF CLUSTER TOOLS .....	11
4.1 Modeling Resource-Oriented Model .....	11
4.2 Finite-Capacity Petri Net .....	14
4.3 Notations .....	14
4.3.1 Places .....	14
4.3.2 Transitions .....	16
4.3.3 Anti-Places .....	16
4.3.4 Time Durations .....	16
5 SIMULATION AND ANALYSIS .....	18
5.1 Simulation and Analysis Overview .....	18
5.2 Steady-State .....	19

**TABLE OF CONTENTS**  
**(Continued)**

<b>Chapter</b>	<b>Page</b>
5.3 Initial Transient Process .....	26
5.4 Final Transient Process .....	31
6 CONCLUSION .....	34
APPENDIX A GRAPHS & PROGRAM .....	36
APPENDIX B TINA INSTRUCTIONS .....	41
APPENDIX B CPN TOOLS INSTRUCTIONS .....	42
REFERENCES .....	43

## LIST OF TABLES

<b>Table</b>	<b>Page</b>
4.1 Time Durations For Different Activities In The Cluster Tool.....	17

## LIST OF FIGURES

<b>Figure</b>	<b>Page</b>
2.1 Petri net example with two states on making lemonade.....	5
2.2 Anti-place example.....	8
3.1 Quad-station cluster tool for modeling.....	10
4.1 Steady-state Petri net for a single process module.....	12
4.2 Steady-state Petri net for a single process module using a static recipe.....	13
5.1 Steady-state Petri net.....	20
5.2 Steady-state Petri net only for PM2 and PM4.....	21
5.3 Steady-state Petri net only for PM1 and PM5.....	24
5.4 Analysis of steady-state Petri net in TINA.....	26
5.5 Initial transient process Petri net.....	29
5.6 Analysis of initial transient process Petri net in TINA.....	31
5.7 Final transient process Petri net.....	32
5.8 Analysis of final transient process Petri net in TINA.....	33
A.1 Petri net graph memory usage.....	40

## LIST OF VIDEOS

Figure	Page
5.1 Clip of video for steady-state simulation using CPN Tools available at <a href="https://www.youtube.com/watch?v=BX6NO9hwBk4">https://www.youtube.com/watch?v=BX6NO9hwBk4</a> .....	22
5.2 Clip of video for steady-state simulation of only PM2 and PM4 using CPN Tools available at <a href="https://www.youtube.com/watch?v=ghXFskXIHF8">https://www.youtube.com/watch?v=ghXFskXIHF8</a> .....	23
5.3 Clip of video for steady-state simulation of only PM1 and PM5 using CPN Tools available at <a href="https://www.youtube.com/watch?v=IwZA9bdtHsI">https://www.youtube.com/watch?v=IwZA9bdtHsI</a> .....	23
5.4 Clip of video for steady-state simulation of a standard recipe with delayed tokens using CPN Tools available at <a href="https://www.youtube.com/watch?v=GHcqyrRj64A">https://www.youtube.com/watch?v=GHcqyrRj64A</a> .....	25
5.5 Clip of video for steady-state simulation of a static recipe with priority transitions using CPN Tools available at <a href="https://www.youtube.com/watch?v=DSTGq8cbDKw">https://www.youtube.com/watch?v=DSTGq8cbDKw</a> .....	25
5.6 Clip of video for initial transient process with a different number of wafers and cases simulation using CPN Tools available at <a href="https://www.youtube.com/watch?v=jQFtO3G_wu4">https://www.youtube.com/watch?v=jQFtO3G_wu4</a> .....	30
5.7 Clip of video for final transient process simulation using CPN Tools available at <a href="https://www.youtube.com/watch?v=xnXOXkpKfcg">https://www.youtube.com/watch?v=xnXOXkpKfcg</a> .....	33

# **CHAPTER 1**

## **INTRODUCTION**

The semiconductor industry is highly competitive, and with the recent chip shortage, the throughput of wafers has become more important than ever. One of the tools that the industry has widely deployed is to use quad-station modules instead of the traditional single-station modules that allow for higher throughput and better wafer consistency by processing multiple wafers simultaneously. The industry trend is to use multiple transfer chamber robots to stack the quad-station modules in a series, particularly for etch products. In this work, the quad-station cluster tool wafer movement is modeled by using Petri nets. As a process-bounded system, such a cluster tool is described via timed and colored Petri nets. Its detailed analysis and simulations are conducted in this thesis work.

### **1.1 Overview**

This thesis work models a production semiconductor quad-station process module cluster tool by using Petri net. The model is then analyzed for reachability, boundedness, liveness, and reversibility by using TINA and simulated for wafer movements by using CPN Tools. This is an exploration to see if Petri net scheduling can be pragmatically implemented in a production environment. Currently, the industry uses a heuristic rule-based approach to schedule cluster tools as it is flexible, extensible, and provides near or quasi-optimal solutions. However, it relies on a developer's experience and creativity. Any mature ruleset for a new tool configuration is derived through trials and errors and, as such, compromises the quality and reliability of the scheduler component in its initial stage. As tool complexity increases, the complexity of the scheduler component becomes higher and higher. Using



the Petri Nets is not an outright rejection of a heuristic approach. In fact, Petri Nets can be used in conjunction with various heuristic approaches, according to [6] [14].

## **1.2 Cluster Tool Background**

Cluster tools process wafers for the fabrication of micro-electric components using automated robotic manufacturing systems. [9] [10] [15] [19] [20] Cluster tools are systems that allow for the transfer of wafers process modules (PMs), buffers, and loadlocks. They can be used for the parallel process of multiple wafers for improved throughput. The general configuration of a cluster tool includes multiple PMs, one or more transfer chambers, and loadlocks. Beyond the loadlocks, the system is operated under a vacuum environment using one or more robotic arm(s) to prevent contamination or other undesired reactions.

Until recently, due to limited floor space in fabs, cluster tools tend to have no more than six process modules, which tend to be single-station process modules. However, with the need for higher productivity and throughput, quad-station process modules are being introduced with some cluster tools supporting up to ten process modules.

In a typical operation, unprocessed wafers are transferred to the cluster tools in FOUPs (Front Opening Unified Pods). FOUPs generally hold 25 wafers. Most cluster tools have the capacity to handle multiple FOUPs. Upon the arrival of FOUP to a cluster tool, the unprocessed wafers are transferred by the atmosphere robotic arm(s) into the loadlock, which are airlocks that reduce the pressure to vacuum for the transfer chamber and PM operations. In some systems, aligners are used to align the wafers to the proper orientation for processing purposes.

Once the wafers are in the transfer chamber, they are quickly moved to the appropriate PM for the recipe that is being used. Some recipes call for PM revisits, meaning that the wafer will visit a specific PM more than once. Once processing is complete, the wafers are removed from the system through the loadlocks back to the FOUPs. In some processes, there are time constraints on how long a wafer can remain in the PM, and there are also some requirements for stay uniformity for quality control purposes.

Depending on the process, PMs generally need to be cleaned after each operation, with non-zero cleaning time. Quad-station modules have an advantage here as the PM only needs to be clean after processing four wafers instead of one. Quad stations also break down the operation into four sub-operations, improving the uniformity of wafers.

In the process, the graph aspect of the reachability graph is also explored and can be found in the Appendix. The rest of the document is organized as follows. Chapter 2 describes cluster tools in general and introduces why quad-station modules are being used. Chapter 3 goes over the basic definitions of Petri nets. Chapter 4 is the description of the cluster tool being modeled. Chapter 5 is the Petri net model of the cluster tool. Chapter 6 is the simulation and analysis of the Petri net model. And chapter 7 concludes the thesis.

## CHAPTER 2

### PETRI NET OVERVIEW

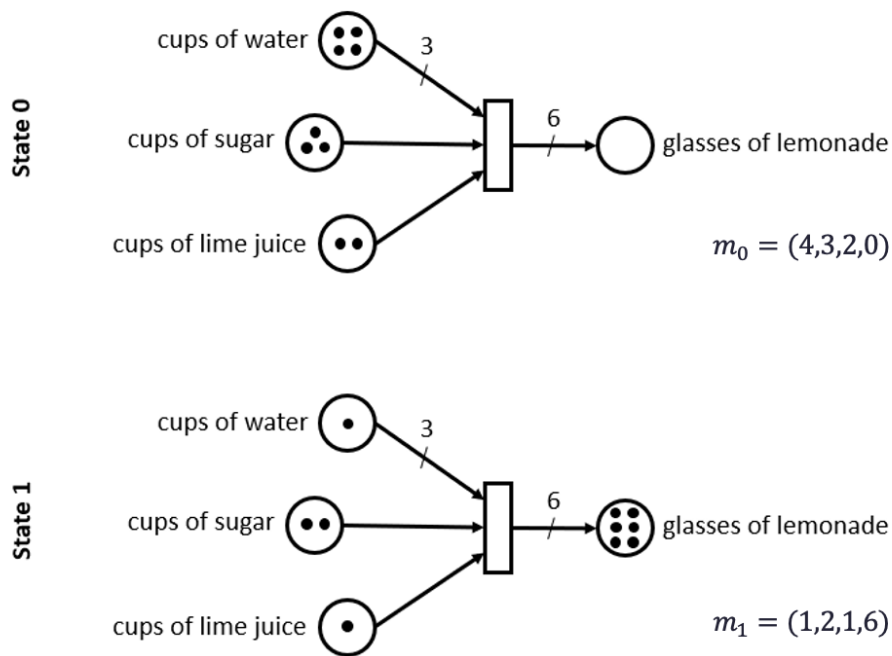
Petri nets are one of the modeling tools available to simulate, evaluate, and model different mechanisms, operations, and procedures [7] [8]. Petri Nets bring the mathematical and graphical aspects of modeling different scheduling problems in event-driven systems.

#### 2.1 Petri Net Basics

It supports the representation of asynchronous, sequential, and concurrent operations through the structure and dynamics of discrete event systems. [5] We can model the flow of information and controls in systems using Petri nets. And most importantly, we can use the analytical properties of Petri net, such as reachability, boundedness/safeness, liveness, and reversibility, to understand the problems better. [5] Petri nets are directed bipartite graphs with places and transitions with static (inputs and outputs) and dynamic (executions and events) properties. There are many variants of the "basic" Petri nets, which include extensions and abbreviations. Some variants of notes include edge types, which allow for extended control of the systems; time, which allows for performance analysis; color, which allows for model attributes; and hierarchy, which allows for the structuring of models. [13]

As we can see in Figure 2.1, the concept of Petri net is easy to follow. Let's say you have the following items: 4 cups of water, 3 cups of sugar, and 2 cups of lime juice. We want to make lemonade and then serve the lemonade in glasses (not equivalent to a cup) that you recently bought. According to a simple google search, the lemonade recipe calls for 3 cups of water, 1 cup of sugar, and 1 cup of lime juice. Once you make lemonade, you

will get a total of 6 glasses of lemonade. All this information can be seen in State 0 of Figure 2.1. The tokens (dots) that are in the input places (circles) to the transition (rectangle) tell us how many resources we have. Based on the arc weights, we can evaluate if we can even attempt to make lemonade (this is called whether the transition can be enabled or not). We can say the transition is enabled if we have enough resources to make lemonade. That does not mean we will make lemonade. It just means that we have the resources to make lemonade. If we did make lemonade (that is firing the transition), we would get six glasses of lemonade. State 0 is the state prior to firing the transition, meaning the event of making lemonade, and State 1 is the state after firing the transition. In State 1, we can see that we have used up 3 cups of water, a cup of sugar, and lime juice each. We now also have six glasses of lemonade.



**Figure 2.1** Petri net example with two states on making lemonade.

## 2.2 A Marked Petri Net Formal Definition

The general marked Petri Net [11] [17] [21] can be described formally as follows. A marked Petri net  $(PN) Z = (P, T, I, O, m)$  is a five-tuple where:

- $P = \{p_1, p_2, \dots, p_n\}$ .  $n > 0$ , is a finite set of places pictured by circles;
- $T = \{t_1, t_2, \dots, t_n\}$ .  $n > 0$ , is a finite set of places pictured by bars, with  $P \cup T \neq \emptyset$  and  $P \cap T = \emptyset$ ;
- $I: P \times T \rightarrow N$ , is an input function that defines the set of directed arcs from  $P$  to  $T$  where  $N = \{0, 1, 2, \dots\}$ ;
- $O: P \times T \rightarrow N$ , is an output function that defines the set of directed arcs from  $T$  to  $P$ ;
- $m: P \rightarrow N$ , is a marking whose  $i$  th component represents the number of tokens in the  $i$  th place. An initial marking is denoted by  $m_0$ . Tokens are represented by dots.

The four-tuple  $(P, T, I, O)$  is called a Petri Net structure that defines a directed graph structure.

Enabling:  $t$  is enabled at marking  $m$

- if  $\forall p \in P, m(p) \geq I(p, t)$ .

Note: If  $I(p, t) = 0$ , the above equation holds regardless of  $m(p)$ . If  $t$  has  $k$  input places,  $k$  such relations must hold simultaneously.

Enabling means that if a place belongs to the Petri net in question, there need to be enough tokens in the place that is required by the transition's input. For example, if the transition input requires two tokens, the place needs to have at least two tokens to enable the transition. The most important thing to note here is that just because a transition is enabled does not mean it would necessarily fire. It just has the potential to fire.

Firing: An enabled transition  $t$  at  $m$  can fire, yielding a new marking  $m'$  such that

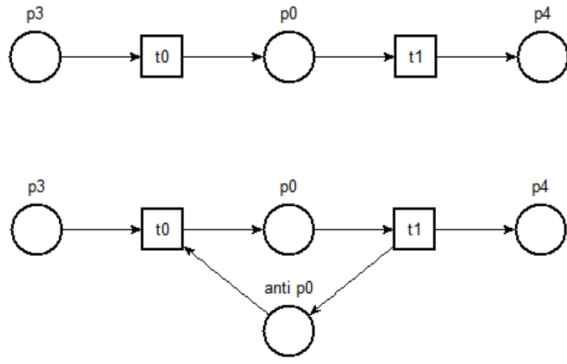
- $\forall p \in P, m'(p) = m(p) - I(p, t) + O(p, t)$

This is applicable to all Petri nets.

The firing means that when an enabled transition fires, it will create a new state (m'). The change between the old state (m) and the new state (m') is for each of the places that are related to this particular transition, one must take away the tokens that are required for firing the transition and add the tokens that are the result of the firing of the transition. This means that when a transition has been enabled, the transition can fire.

### 2.3 Anti-Place Definition

The definition of an anti-place [2] is such that it has a corresponding original place with two transitions that are inputs and outputs for the place. As shown in Figure 2.2, we have  $p_0$ , place, and the anti  $p_0$ , anti-place. The transition  $t_0$  has outputs going into  $p_0$ , and the transition  $t_i$  has inputs from  $p_0$  in Petri net without anti-place. To create an anti-place and impose a capacity limit on  $p_0$ , we must first create an anti-place, anti  $p_0$ , then connect arcs from anti  $p_0$  to  $t_0$  and from  $p_i$  to anti  $p_0$ . The key thing here is that the weight of the arc goes from anti  $p_0$  to  $t_0$  must match the arch going from  $t_0$  to  $p_0$ , and the arc goes from  $t_0$  to anti  $p_0$  must match the arch going from  $p_0$  to  $t_i$ . They are accounting for the tokens going in and out of  $p_0$ . Then, we must place tokens in anti  $p_0$  such that  $M(\text{anti } p_0) = K(p_0) - M(p_0)$ , meaning the finite capacity we are imposing on  $p_0$  is equal to the total number of tokens in  $p_0$  and anti  $p_0$ . (If we have colored tokens going from  $t_0$  to  $p_0$  and to  $t_i$ , we must make sure that the tokens go from and to anti  $p_0$  are either uncolored or of a different color.) By adhering to the total number of tokens, there can only be a maximum of  $K(p_0)$  tokens at  $p_0$ . We can use this approach to limit the capacity of not just a single place but also multiple places that are capped by a pair of transitions.



**Figure 2.2** Anti-place example.

## CHAPTER 3

### CLUSTER TOOL OVERVIEW

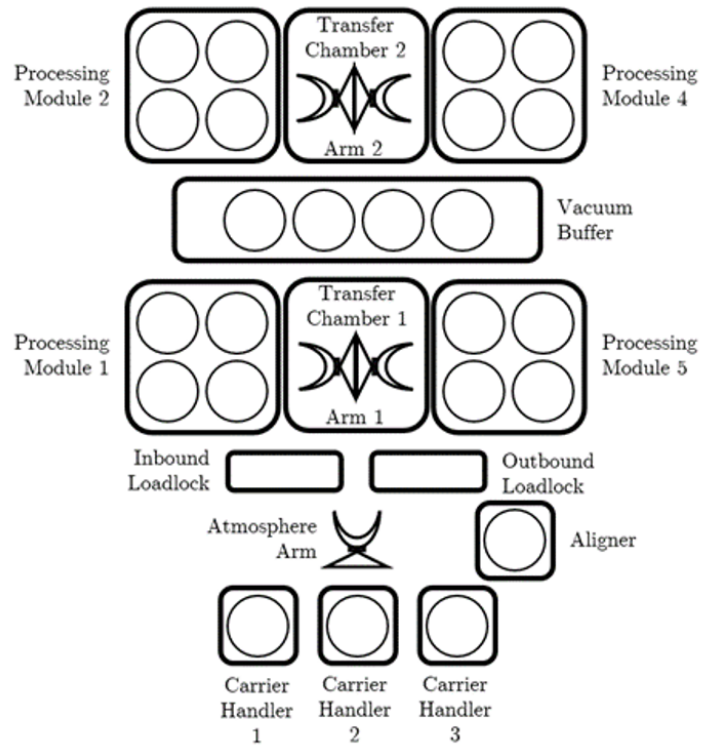
The cluster tool being modeled is a semiconductor production unit used for 3D NAND production by various storage manufacturers.

#### 3.1 Description of Cluster Tool

As shown in Figure 3.1, it has four quad-station process modules, two transfer chambers with dual arms with a single hinge in each, a buffer with five slots for wafers, an atmosphere arm that operates between carrier handlers and the airlocks, and an aligner. In this study, we assume that the scheduling time is process-bound, the wafers have a single recipe, and all PMs run the same recipe. The raw wafers from the carrier handlers are picked up by the atmosphere arm and are aligned prior to being inserted into the loadlocks. Once the wafers enter the vacuum chamber, they visit a single PM. Depending on the PM being visited, the path taken by the wafer differs in the initial and final transient processes. However, during the steady-state, all wafers travel the same path through the vacuum buffer; that is, the wafer travels as follows: Carrier Handler -> Aligner -> Loadlock -> Vacuum Buffer -> PM -> Vacuum Buffer -> Loadlock -> Carrier Handler. The single-arm robot handles the transfer operation between the carrier handler and the loadlock, including the aligner. As for the transfer operation between the loadlock, PMs, and the vacuum buffer, they are handled by the dual-arm robots in transfer chambers 1 and 2. Transfer chamber 2 is responsible for transfer operations between the vacuum buffer and either PM<sub>2</sub> or PM<sub>4</sub>, while transfer chamber 1 is utilized for transfer operations between the loadlocks, vacuum buffer, and either PM<sub>1</sub> or PM<sub>5</sub>. All the PMs are quad-station modules (QSM), and



the vacuum buffer has a capacity for five wafers. The aligner can align only one wafer at a time.



**Figure 3.1** Quad-station cluster tool for modeling.

## CHAPTER 4

### PETRI NET MODELS OF CLUSTER TOOLS

In this section, we model the cluster tool described in Chapter 3, which is a cluster tool with quad-station processing modules.

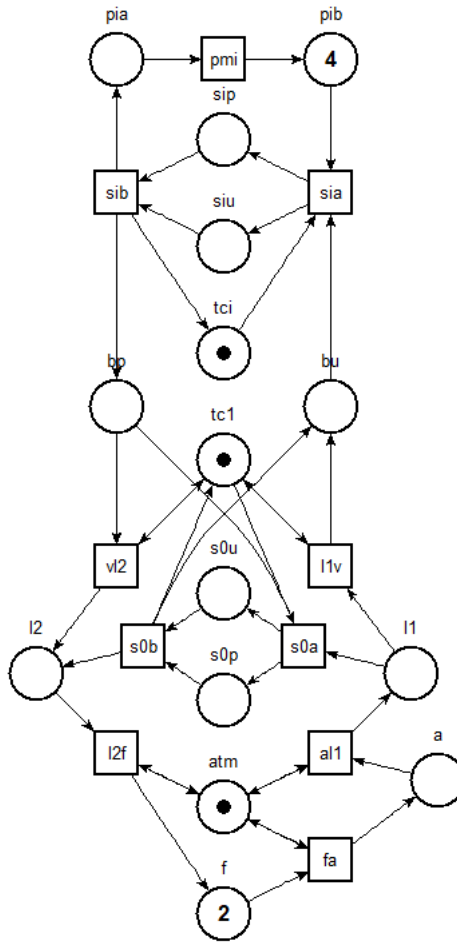
#### 4.1 Modeling Resource-Oriented Model

Modeling using Resource-Oriented Petri net [13] is split into three categories:

1. steady-state
2. initial transient process (batch begin)
3. final transient process (batch end).

For this particular model, the process is neither multi-visit nor repeating, meaning the wafer only has to visit one PM and exits after the process. Hence, the Petri net combines four resource-oriented Petri nets for each PMs with a cyclic process. The four Petri nets, when combined, share resources, including the robotic arms, vacuum buffer, loadlocks, and atmosphere components. (Please see below for "Notations.") In this single PM steady-state Petri net, as you can see in Figure 4.1, at  $PM_i$ , there is a swap operation [5] of wafers between  $PM_i$  and the vacuum buffer whereby an unprocessed wafer is picked up from the vacuum buffer, then a processed wafer is removed from the  $PM_i$ , and an unprocessed wafer is inserted into the  $PM_i$ , after which the processed wafer is placed into the vacuum buffer. There are two different operations with the loadlock: a swap operation and a single wafer operation. During the swap operation, a processed wafer is placed into the outbound loadlock, and an unprocessed wafer is picked up from the inbound loadlock. There is also an independent process of transferring processed wafers to the outbound loadlock and

unprocessed wafers from the inbound loadlock. In addition, the atmosphere arm handles the transfer of wafers between the loadlocks, aligner, and the FOUP. This description is for the standard recipe process in which a single wafer is moved into the PM at a time, and the individual wafer(s) are processed separately.

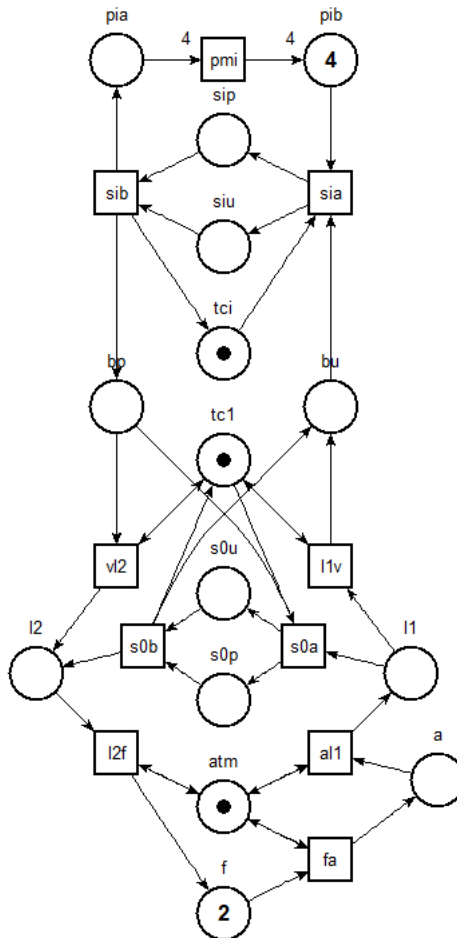


**Figure 4.1** Steady-state Petri net for a single process module.

In contrast, Figure 4.2 describe the model in which we can use the static recipe process that processes four wafers at a time. One thing we should note here is that we are only using one token for the robot arms ( $tc_1$  and  $tc_i$ ), which generally is associated with single-arm robots. However, in our case, the dual-arm robots are used in single operations to either swap or simple transfers. Hence, the use of a single token for the robot arms. In

this Petri net model, the weight of the arcs for the transfer operation with  $PM_i$  is 1, meaning we are just modeling for a single robot with dual arms with dedicated arms for processed and unprocessed wafers. To change this into a single robot with quad arms with dual processed and unprocessed arms, we can simply change the transfer operation arc weights and swap place capacity to 2. For such a change, we can also assume that the aligner can align two wafers simultaneously for model simplification.

Note: All Petri net analyses are done in TINA (Time petri Net Analyzer), and visual simulations are done in CPN Tools.



**Figure 4.2** Steady-state Petri net for a single process module using a static recipe.

## 4.2 Finite-Capacity Petri Net

A finite capacity Petri net  $(PN)Z = (P, T, I, O, m, K)$  is a six tuple where, in addition to the same five definitions of the tuples as above:

$K: P \rightarrow N - \{0\}$  is a capacity function where  $K(p)$  represents the maximal number of tokens that  $p$  can hold

Enabling of a finite capacity Petri net:  $t$  is enabled at marking  $m$ :

- If  $\forall p \in P$ , the following two conditions must hold true:
  - $m(p) \geq I(p, t)$  and
  - $K(p) \geq M(p) - I(p, t) + O(p, t)$

The firing definition remains the same as the prior definition above.

The main difference between a finite capacity Petri net and a Marked Petri net is that places that have a finite limit have a capacity limit for the places, meaning if the change in the number of tokens, by firing a transition, exceeds the limit of the place, the transition would not be enabled to begin with.

## 4.3 Notations

Below are the notations used for the model, including its places, transitions, and anti-places, as well as their conditions, capacities, and some definitions. The last sub-section also includes the table for the time durations of activities and operations.

Let  $\mathbb{N}_n = \{1, 2, 4, 5\}$  and  $\Omega_{n\_n} = \mathbb{N}_n \cup \{0\}$ .

### 4.3.1 Places

Places are represented by circles. They model resources, buffers, conditions, or states.

- $f$ : FOUP/Carrier Handler
- $a$ : Aligner with  $K(a) = 1$
- $atm$ : Atmosphere Arm
- $l_1$ : Inbound Loadlock  $K(l_1) = 1$
- $l_2$ : Outbound Loadlock  $K(l_2) = 1$
- $tc_1$ : Transfer Chamber 1 (with dual-arm robot) with  $K(tc_1) = 1$
- $tc_2$ : Transfer Chamber 2 (with dual-arm robot) with  $K(tc_2) = 1$
- $b_u$  and  $b_p$ : These are virtual vacuum buffers for unprocessed and processed wafers, respectively, where  $K(b_u) + K(b_p) = 5$ . They are, in fact, a single unit with five wafer capacities.
- $p_{ia}$  models the pre-processing condition at  $pm_i, i \in \mathbb{N}_n$ , where  $\mathbb{N}_n = \{1,2,4,5\}$  with  $K(p_{ia}) = 4$
- $p_{ib}$  models the condition post-processing conditions at  $pm_i, i \in \mathbb{N}_n$ , where  $\mathbb{N}_n = \{1,2,4,5\}$  with  $K(p_{ib}) = 4$
- $s_{iu}$  models the movement of the unprocessed wafer during the swap operation, as well as the wait time, at  $pm_i, i \in \Omega_n$ , where  $\mathbb{N}_n = \{0,1,2,4,5\}$  with  $K(s_{iu}) = 1$
- $s_{ip}$  models the movement of the processed wafer during the swap operation, as well as the wait time, at  $pm_i, i \in \Omega_n$ , where  $\mathbb{N}_n = \{0,1,2,4,5\}$  with  $K(s_{ip}) = 1$

### 4.3.2 Transitions

Transitions are represented by rectangles. They model events, transformations, or transportations.

- $s_{i1}$  models the beginning of the swap process between a processed and unprocessed wafer at  $pm_i, i \in \Omega_n$ , where  $\mathbb{N}_n = \{0,1,2,4,5\}$
- $s_{ib}$  models end of the swap process between a processed and unprocessed wafer at  $pm_i, i \in \Omega_n$ , where  $\mathbb{N}_n = \{0,1,2,4,5\}$
- $pm_i$  models the processing of wafers at  $pm_i, i \in \Omega_n$ , where  $\mathbb{N}_n = \{1,2,4,5\}$

- $l_1v$  and  $vl_2$ : models the movement of a wafer from loadlock to the vacuum buffer and vice versa
- $al_1$ : models the movement of a wafer from the aligner to the loadlock
- $fa$ : models the movement of a wafer from the FOUP to the aligner
- $l_2f$ : models the movement of a wafer from the loadlock to the FOUP

### 4.3.3 Anti-Places

Anti-Places are capacity control for corresponding places. The anti-places are not shown in the Petri net model images. But only in the simulation videos, as the CPN Tools does not support finite capacity Petri nets.

- $ta$ : anti-place for aligner
- $tl_1$ : anti-place for loadlock 1
- $tl_2$ : anti-place for loadlock 2
- $tb$ : anti-place for the combined places of  $b_u$ , buffer for unprocessed wafers, and  $b_p$ , buffer for processed wafers

### 4.3.4 Time Durations

The time durations for different activities in the cluster tools are defined as follows in table 4.1. It includes different robot activity times and the corresponding wait times, as well as other components and their processing time and their wait times.

**Table 4.1** Time Durations For Different Activities In The Cluster Tool

Symbol	Transition or Place	Action	Duration allowed
$\mu$		Any robot arm movement	
$\sigma$	$s_{ia}, s_{ib}$	Simple swap operation at PM <sub>i</sub>	
$\omega_{is}$	$s_{iu}, s_{ip}$	Robot wait time at swap operation at PM <sub>i</sub>	$[0, \gamma]$
$\omega_{ip}$	$p_{ia}$	Robot wait time at the beginning of swap operation at PM <sub>i</sub>	$[0, \infty]$
$\omega_{iz}$	$p_{ib}$	Robot wait time at the end of the swap operation	$[0, \infty]$
$\omega_{il}$	$l_1v, vl_2, al_1, l_2f$	Robot wait time at loadlock	$[0, \infty]$
$\omega_{ia}$	$al_1, fa$	Robot wait time at aligner	$[0, \infty]$
$\omega_{if}$	$fa$	Robot wait time at FOUP	$[0, \infty]$
$\tau_i$	$pm_i$	Wafer being processed and waiting in PM <sub>i</sub>	$[a_i, a_i + \delta_i]$
$\lambda$	$l_1, l_2$	Loadlock time	
$\rho$	$a$	Aligner time	
$\zeta$	$pm_i$	PM <sub>i</sub> processing completion time	
$\zeta_{ib}$	$pm_{ib}$	Processing completion time at the first PM used in the initial transient process	
$\pi$		$(4\mu + \lambda + \omega_{il} + \rho + \omega_{ia} + \omega_{is} + \omega_{ip} + \omega_{iz})$ time for each wafer to get from FOUP to PM <sub>i</sub>	
$\varepsilon$	$pm_i$	PM <sub>i</sub> processing time	



## CHAPTER 5

### SIMULATION AND ANALYSIS

In this section, the Petri net model described in Chapter 4 is analyzed and simulated using TINA and CPN Tools, respectively.

#### 5.1 Simulation and Analysis Overview

For analysis, TINA (Timed petri Net Analyzer) is used specifically to check for reachability, boundedness, liveness, and reversibility. The reachability graph of a Petri net provides a fundamental basis for the dynamic properties of a system. It allows us to determine whether our modeled system can attain a specific state due to a required functional behavior with a single or a sequence of events. It can also be used to check if there are any deadlocks in the system. In a bounded PN, the reachability problem is decidable, but the complexity is exponential. For boundedness, a Petri net is considered  $k$ -bounded or bounded if the number of tokens in each of the places is less than or equal to  $k$ , which is a finite value. The boundedness of the system allows us to determine if there are overflows in the system. A system's liveness tells us if there are any deadlocks in the system. If a system is not live, it means that there are deadlocks. However, there are different levels of liveness with regard to how many times a transition in a Petri net can fire, which ranges from 1 to  $k$ , where  $k$  = every iteration. The reversibility of a PN tells us that the modeled system can go back to the original state of the system prior to firing of transition using a series of firings.

CPN Tools is one of the very useful tools to simulate Petri nets. CPN Tools can work with colored and timed Petri nets. It also uses a dialect of SML (Standard Markup

Language) to add additional functionality and control. However, it is no longer maintained, and some existing bugs hinder its use. Yet, for certain usages, omitting certain functionalities, it can still be a great tool. It is also the tool with the best documentation, although it can be outdated and irrelevant. The developers have moved on to Access/CPN tool, which is another Petri net tool based on CPN Tools but has significantly less functionality. CPN Tools was used as a visual simulator of wafer movement.

The wafers in the simulations are shown with changing green rectangles with rows of values. Each row has three distinct pieces of information:

- quantity of wafers of specific status and availability
- status of the wafer, whether it is unprocessed or processed (shown with either 'u' or 'p')
- time, in seconds, at which the wafer is available/ready

The format of the notation is as follows, split by ‘’ and '@.'

Quantity'WaferStatus@Time

In some cases, the transition  $x_0$  is used to simulate the cyclic nature of the system.

Each timed transition has a set constant delay specified by the prefix@+. There are additional places with finite tokens and arcs to model to finite capacity. There are also some inhibitor arcs that model a form of control.

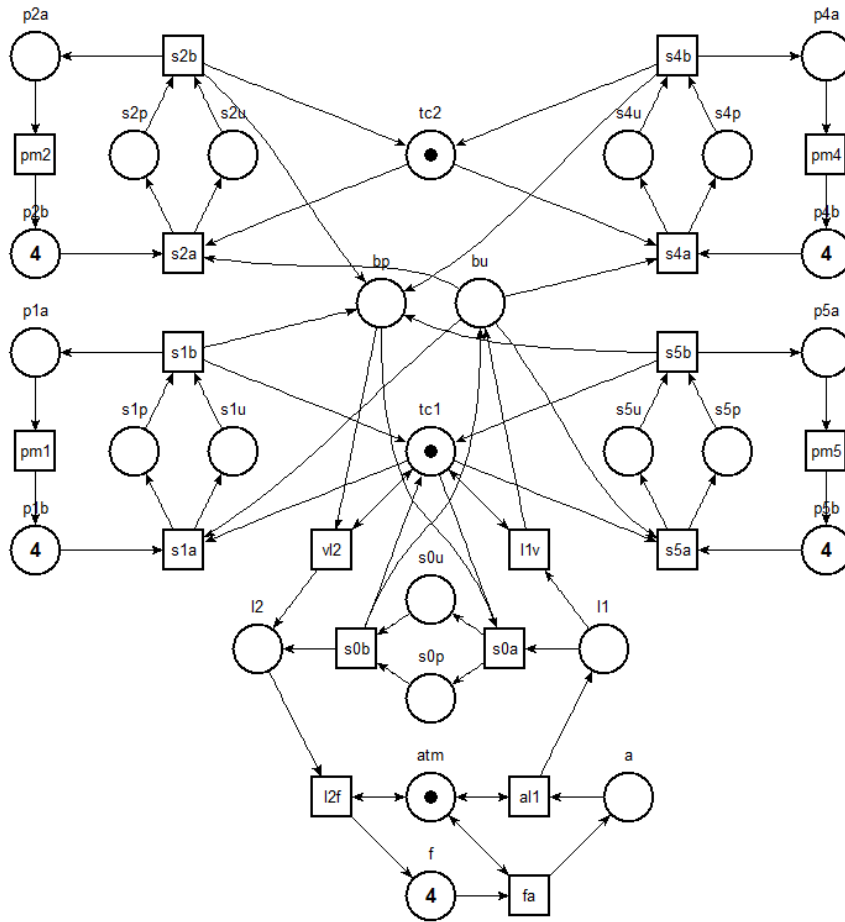
## 5.2 Steady-State

The steady-state is defined as meeting the following conditions in the Petri net:

- $M(p_{ib}) = K(p_{ib}) = 4, i \in \mathbb{N}_n$ , where  $\mathbb{N}_n = \{1,2,4,5\}$
- $M(tc_i) \geq 1, i \in \mathbb{N}_t$ , where  $\mathbb{N}_t = \{1,2\}$
- $M(atm) = M(ta) = 1$

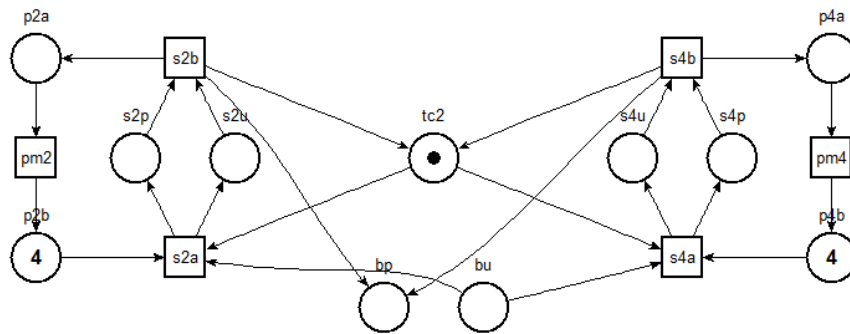
- $M(tl_i) = 1, i \in \mathbb{N}_t, \text{ where } \mathbb{N}_t = \{1,2\}$
- $M(tb) = 5$

Each iteration of the steady-state starts at  $\zeta_{ib} - \pi$ . There is also an assumption that  $\varepsilon \geq 16\pi$ . The process of the steady-state as defined in the single PM, firing of  $s_{ia}, i \in \mathbb{N}_n$ , where  $\mathbb{N}_n = \{1,2,4,5\}$ , transitions will depend on the availability of the processed wafers from each PM, which in turn is based on the optimized initial transient process, as the system is considered process bounded. This is straightforward for PM<sub>2</sub> and PM<sub>4</sub> as the arm is not shared with any other process. However, for PM<sub>1</sub> and PM<sub>5</sub>, the arm is shared with the transfer operations in which wafers are moved from loadlocks to the vacuum buffer.



**Figure 5.1** Steady-state Petri net.

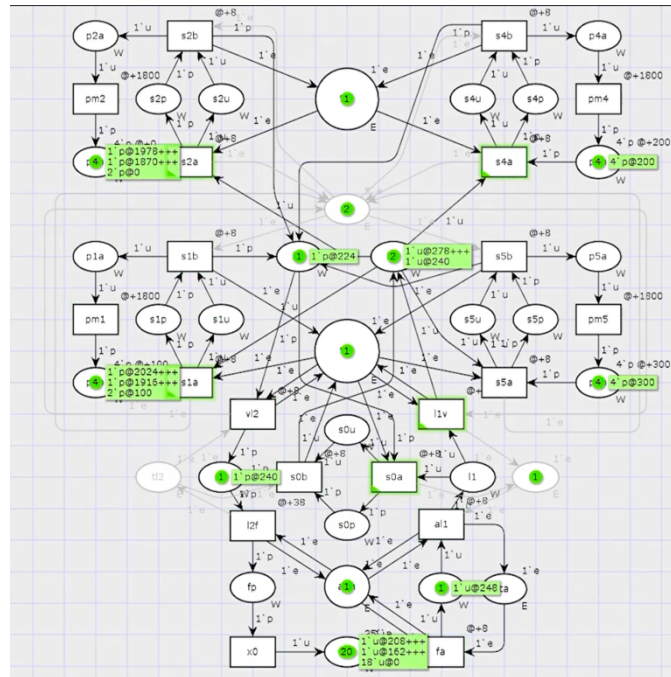
As you can see in Figure 5.2, we can consider  $PM_2$  and  $PM_4$  as a sub-case, and is much simpler to work with. Assuming a constant flow of unprocessed wafers into the vacuum buffer, as the wafer processing of each PM completes, the respective  $s_{ia}$  transition will fire and output wafers into the vacuum wafer, which is  $K(b_pbp) + K(b_u) = 5$ . We can even consider this as a system with two PMs without a buffer for simplicity. The availability of processed wafers will determine the order of the process, which is determined by the initial transient process. In Video 5.1, we see the simulation of the subnet in which only  $PM_2$  and  $PM_4$  are considered, along with the buffers as the input and output of wafers.



**Figure 5.2** Steady-state Petri net only for  $PM_2$  and  $PM_4$ .

There are delays associated with different transitions. Each swap transition has a delay of 8, and the PM transitions have a delay of 1800. The  $x_0$  transition does not have any delays. The key thing to note in this simulation is the availability of wafers at  $PM_4$ . They are not available until time 200. This relative delay in the availability of the wafers at  $PM_4$  and  $PM_2$  allows us to operate this Petri net without any further controls. This setup consistently transfers four wafers to each PM, even though it is a standard recipe model. This would change once we simulate the whole system, and we will observe a behavior

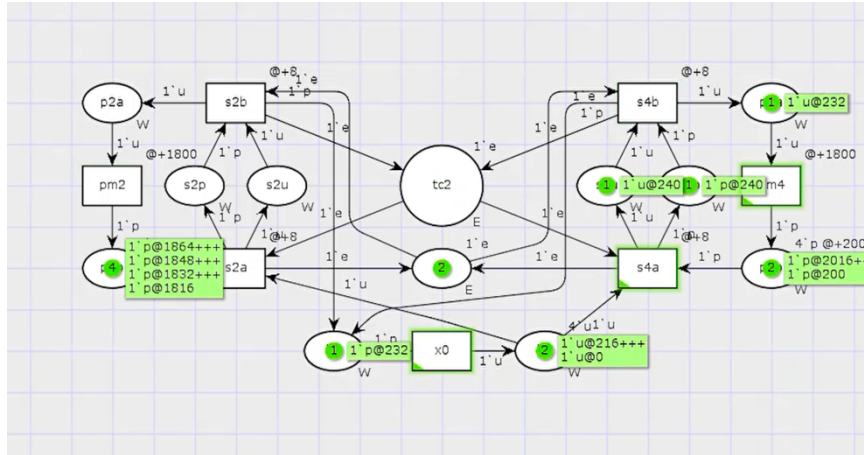
more consistent with the standard recipe, meaning the wafers will be inserted and processed one at a time across different PMs.



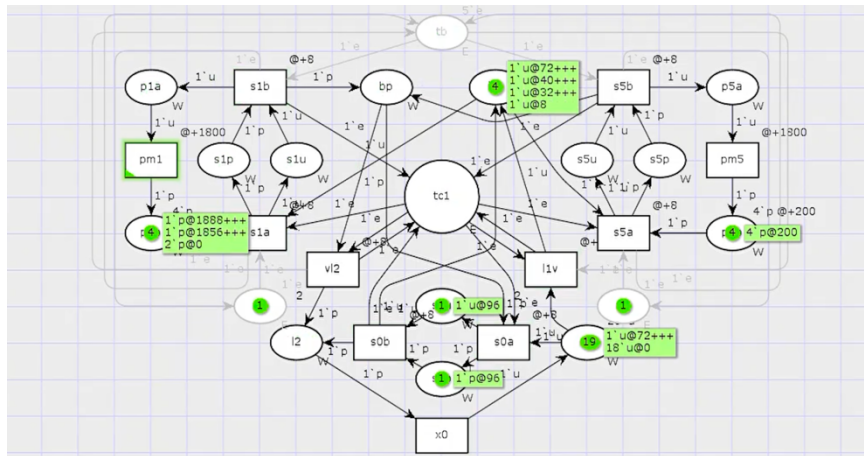
**Video 5.1** Clip of video for steady-state simulation using CPN Tools available at <https://www.youtube.com/watch?v=BX6NO9hwBk4>.

We can also break down the steady-state Petri net into a subnet with only PM<sub>1</sub> and PM<sub>5</sub>. If we look at Figure 5.3, we see that it behaves almost like a system with three PMs, where the buffer acts like a PM that does not require any processing, and wafers are immediately available. In addition, the input of wafers is available from two sources: PM<sub>1</sub> and PM<sub>5</sub> are fed by the buffers, while the buffer is fed by the loadlocks. In Video 5.3, we see a delay of 200 being deployed for PM<sub>5</sub>, which allows us to have this PM without additional control. The swap at loadlock automatically happens whenever there is an unprocessed wafer in the inbound loadlock and a processed wafer in the vacuum buffer. Again, even though this is a standard recipe model, it consistently transfers four wafers to each PM and alternates between PM<sub>1</sub> and PM<sub>5</sub>. When we combine everything and look at

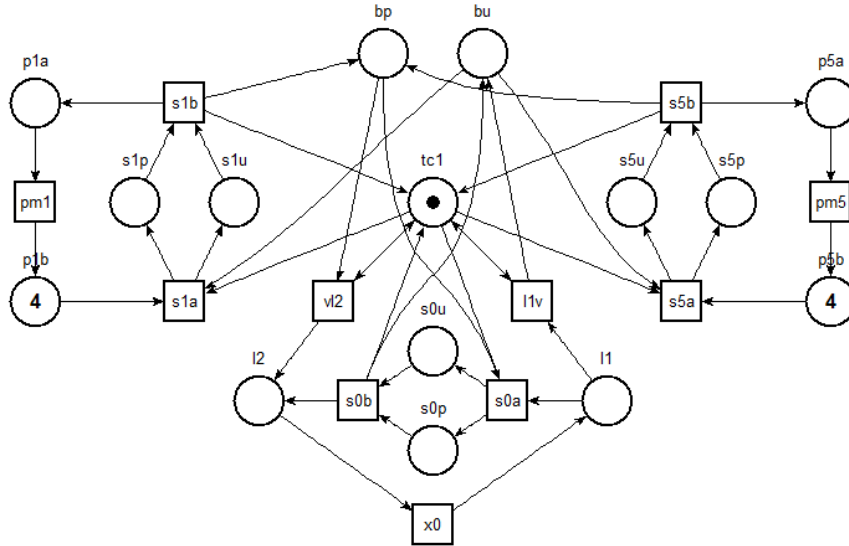
the whole system, the control mechanism required for it gets slightly different based on whether we would like to use a standard or static recipe.



**Video 5.2** Clip of video for steady-state simulation of only PM<sub>2</sub> and PM<sub>4</sub> using CPN Tools available at <https://www.youtube.com/watch?v=ghXFskXIHFE>.

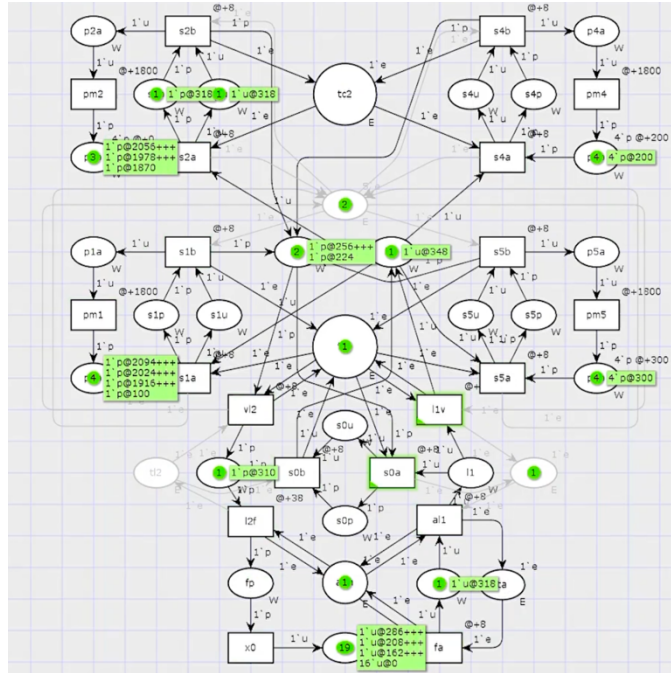


**Video 5.3** Clip of video for steady-state simulation of only PM<sub>1</sub> and PM<sub>5</sub> using CPN Tools available at <https://www.youtube.com/watch?v=IwZA9bdtHsI>.

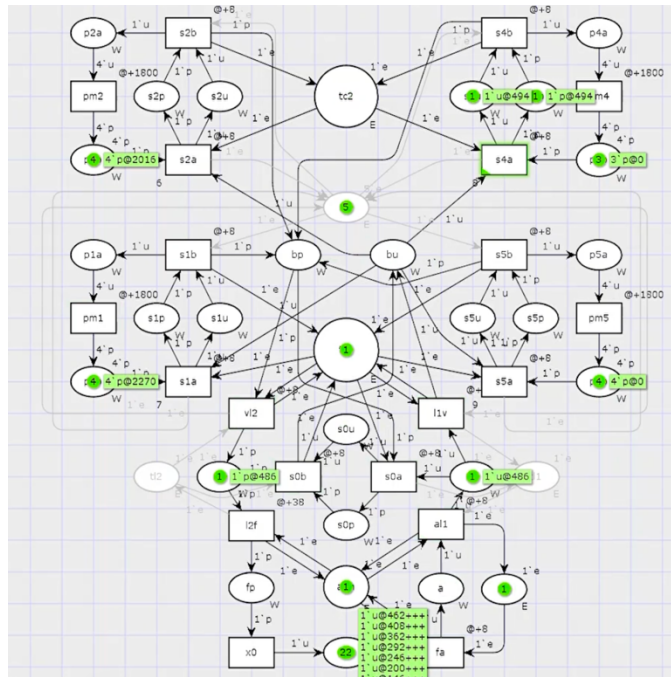


**Figure 5.3** Steady-state Petri net only for  $PM_1$  and  $PM_5$ .

When considering the standard recipe, the first approach of using delays for simulating the availability of the processed wafers from the PMs starts with processing each wafer. For example, all four wafers from  $PM_1$ , in Video 5.4, are available at time 100. However, as it is a free choice Petri net, the system will randomly pick any of the PMs with the available processed wafer to swap. When we look at the static recipe model, using delays for the availability of the wafers does not work as the Petri net behaves in a manner similar to the standard recipe and transfers individual wafers to different PMs based on their availability. However, this is addressed using the transition priority, as shown in Video 5.5. The  $s_{ia}$  transitions have different priorities, meaning since the system is process bounded, the priority transitions behave like a control system in which the wafer transfers are iterated through different PMs in the order of priority.

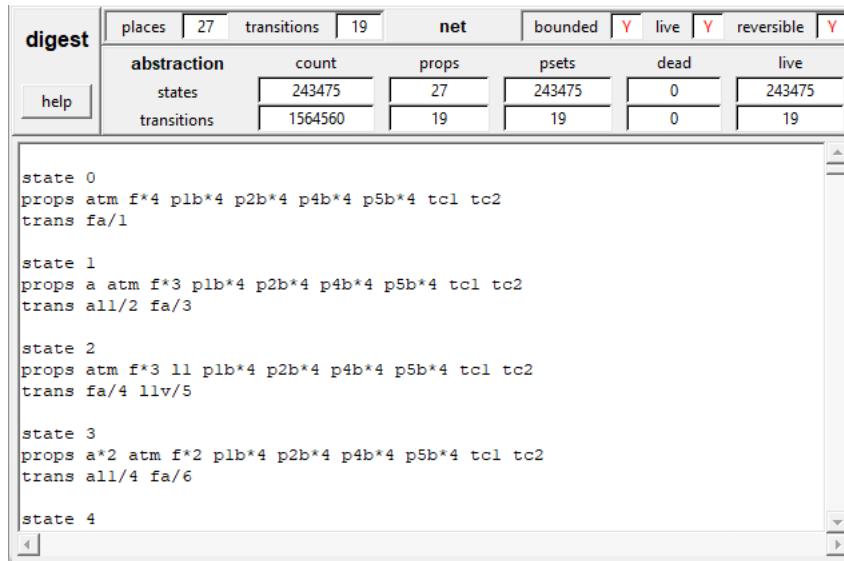


**Video 5.4** Clip of video for steady-state simulation of a standard recipe with delayed tokens using CPN Tools available at <https://www.youtube.com/watch?v=GHCqyrRj64A>.



**Video 5.5** Clip of video for steady-state simulation of a static recipe with priority transitions using CPN Tools available at <https://www.youtube.com/watch?v=DSTGq8cbDKw>.





**Figure 5.4** Analysis of steady-state Petri net in TINA.

As shown in Figure 5.4, the steady-state Petri net is bounded, live and reversible. TINA also generated the reachability states, as shown. There are 243,475 states in total, with 1,564,560 transitions generated from 27 places with 19 transitions. One thing to note here is that the states are being minimized with the use of only four wafers in the FOUP. By increasing the number of wafers in the FOUP, the number of states and transitions will increase while maintaining its properties of boundedness, liveness, and reversibility.

### 5.3 Initial Transient Process

The initial transient process brings the system to meet steady-state conditions. All the PMs prior to the initial transient process are empty. Depending on how many wafers are to be loaded, there are four different cases to consider having an optimal start. Even though the cluster tool has a dual robot arm, for this process, it will behave as though it is operating with a single arm. The objective of the initial transient process is the minimize the time taken to fill the pipeline of wafers into the system to either achieve the steady-state or

initiate the final transient process, depending on the number of wafers that need to be processed. There is a specific strategy to deploy, as there are two distinct durations to get the wafers to different PMs.

The first type of operation for loading wafers is the one involving PM<sub>1</sub> and PM<sub>5</sub>, in which the wafer is:

1. unloaded from the FOUP and loaded into the aligner by the atmosphere robot ( $\omega_{ia} + 2\mu + \omega_{if}$ )
2. unloaded from the aligner, after alignment, and loaded into the inbound loadlock by the atmosphere robot ( $\rho + \omega_{ia} + 2\mu + \omega_{il}$ )
3. unloaded from the inbound loadlock and loaded into PM<sub>1</sub> or PM<sub>5</sub> by the transfer chamber 1 robot ( $\lambda + \omega_{il} + 2\mu + \omega_{ip}$ )

The second type of operation for loading wafers is the one involving PM<sub>2</sub> and PM<sub>4</sub>, in which the wafer is:

1. unloaded from the FOUP and loaded into the aligner by the atmosphere robot ( $\omega_{ia} + 2\mu + \omega_{if}$ )
2. unloaded from the aligner, after alignment, and loaded into the inbound loadlock by the atmosphere robot ( $\rho + \omega_{ia} + 2\mu + \omega_{il}$ )
3. unloaded from the inbound loadlock and loaded into the vacuum buffer by the transfer chamber 1 robot ( $\lambda + \omega_{il} + 2\mu + \omega_{iv}$ )
4. unloaded from the vacuum buffer and loaded into the PM<sub>2</sub> or PM<sub>4</sub> by the transfer chamber 2 robot ( $\omega_{iv} + 2\mu + \omega_{ip}$ )

Ceteris paribus, with other conditions remaining the same, the second type of operation takes  $2\mu$  longer, as  $\omega_{iv} = 0$ , than the first and holds the key to the strategy on how to place wafers when minimizing the initial transient process. Since we are working with quad station modules, depending on the number of wafers involved, the process has repetitions, yet it is not cyclic as it is in a steady-state.

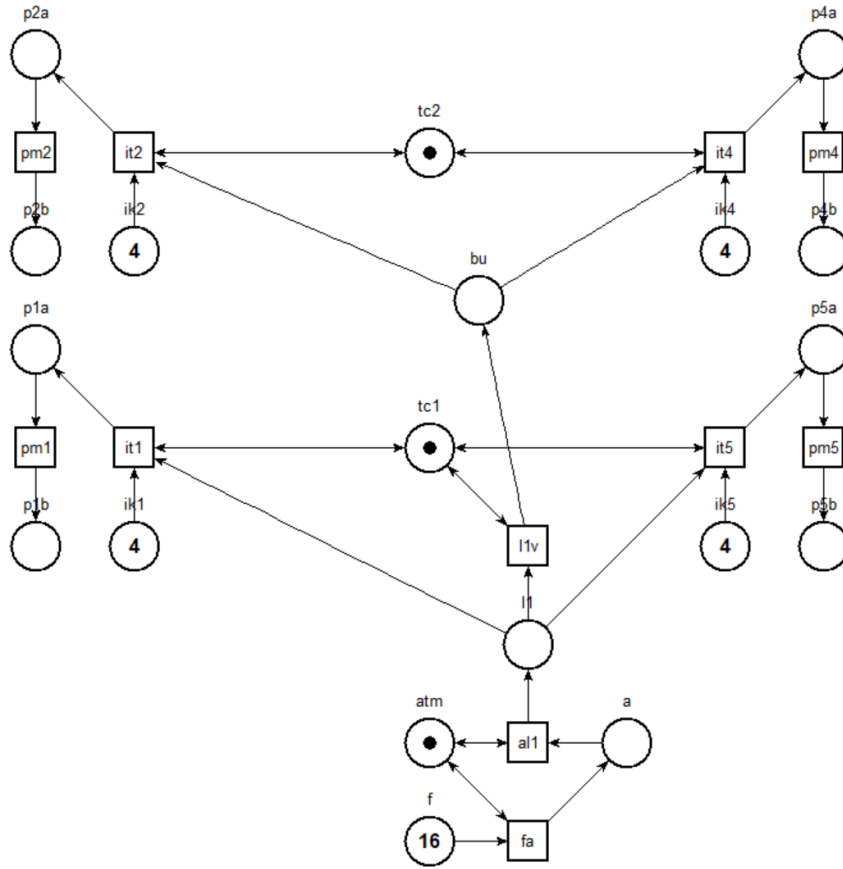
Let the number of wafers to be processed be  $w$ .

Case 1 ( $w \leq 8$ ): The wafers can be loaded into either  $PM_1$  and/or  $PM_5$ , with a priority for filling a single PM first. In this case, the system will not enter the steady-state and proceed to the final transient process after this.

Case 2 ( $8 < w \leq 12$ ): The key here is the  $2\mu$  time difference between the two types of operations. To do so efficiently, we need to schedule the first type of operation (meaning use either  $PM_1$  or  $PM_5$ ) first, followed by the second type of operation (meaning use either  $PM_2$  or  $PM_4$ ). Then alternate back to the first type of operation with the use of either  $PM_1$  or  $PM_5$ . The system will not enter the steady-state in this case either and execute the final transient process.

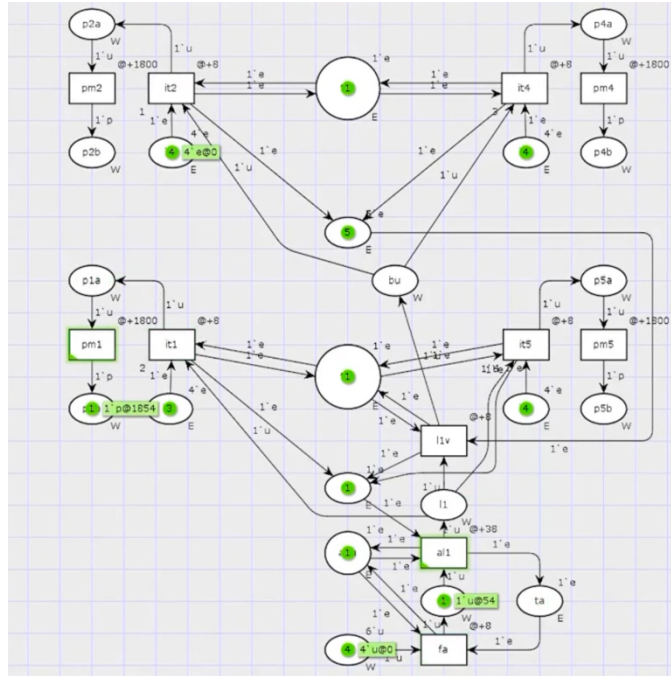
Case 3 ( $12 < w < 16$ ): For this case, there needs to be at least a single alternating event between the two different operations when scheduling the first 2 PMs. Some examples:  $PM_1 \rightarrow PM_2 \rightarrow PM_4 \rightarrow PM_5$  or  $PM_2 \rightarrow PM_1 \rightarrow PM_4 \rightarrow PM_5$ . This will take advantage of the  $2\mu$  time difference. The system will not enter the steady-state in this case either and execute the final transient process afterward.

Case 4 ( $w \geq 16$ ): In this case, all the PMs would be filled. However, as in case 3, we need to alternate between the two operations when scheduling the first 2 PMs. The system will then meet the conditions of the steady-state when all the PMs are filled. It will then be followed by the final transient process when the new wafers run out.



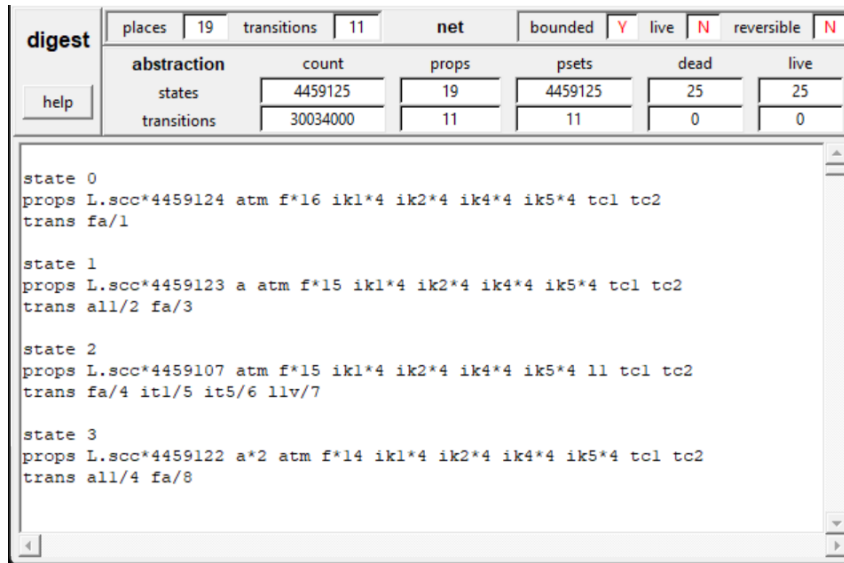
**Figure 5.5** Initial transient process Petri net.

The Petri net being used is a separate and modified version of the standard recipe Petri net. It keeps the wafer input elements. At each PM, we have an additional transition ( $it_i$ ) and an additional place ( $ik_i$ ),  $K(ik_i) = 4$  and  $i \in \mathbb{N}_n$ , where  $\mathbb{N}_n = \{1,2,4,5\}$ . The transition models the transfer operation of wafers to the PM, and the place models the finite capacity control of a quad station module. We can think of this as a sub-system with distinct and compartmentalized behavior for filling the pipeline, mostly to achieve the steady-state, at which point, the steady-state Petri net will be used.



**Video 5.6** Clip of video for initial transient process with a different number of wafers and cases simulation using CPN Tools available at [https://www.youtube.com/watch?v=jQFtO3G\\_wu4](https://www.youtube.com/watch?v=jQFtO3G_wu4).

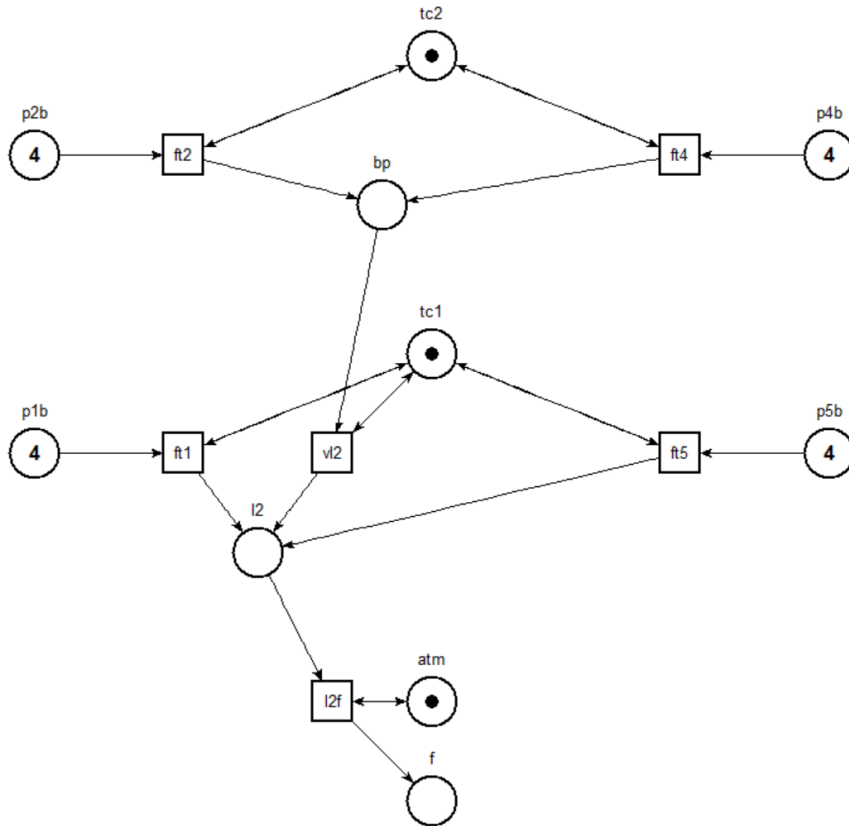
As shown in Figure 5.6, the initial transient process Petri net is bounded but not live nor reversible. It is not expected to be either live or reversible as the task is to fill the system with wafers, and it has a specific end condition at which the net will be switched over to a steady-state or final transient process. TINA also generated the reachability states, as shown. There are 4,459,125 states in total, with 30,034,000 transitions generated from 19 places with 11 transitions. Although it is a simpler net relative to the steady-state, it does have significantly more states in comparison to the steady-state due to the use of more wafers in the FOUP for the analysis.



**Figure 5.6** Analysis of initial transient process Petri net in TINA.

### 5.4 Final Transient Process

The final transient process winds down the system from the steady-state. The process needs to meet the following condition:  $M(f) = M(a) = M(l_1) = M(b_u) = 0$  What this means is that there are no more new wafers in either the FOUP, aligner, inbound loadlock, or buffer. Once this condition is met, for the system, based on the availability of the processed wafers, the wafers would be removed. As we can see in Video 5.7, as the processed wafers become available, each is removed from the system. (In the simulation, there are additional places ( $ft_{ip}$  and  $tft_i$ ) and transitions ( $ft_{it}$ ) to simulate the staggering availability of wafers that will be produced by the steady-state.)



**Figure 5.7** Final transient process Petri net.

As shown in Figure 5.8, the final transient process Petri net is bounded but not live nor reversible. It is also a simpler system relative to the steady-state and the initial transient process. TINA also generated the reachability states, as shown. There are 23,125 states in total, with 104,000 transitions generated from 10 places and 6 transitions.

digest		places	transitions	net	bounded	live	reversible
		10	6		Y	N	N
abstraction		count	props	psets	dead	live	
states		23125	10	23125	1	1	
transitions		104000	6	6	0	0	

```

state 0
props L.scc*23124 atm plb*4 p2b*4 p4b*4 p5b*4 tc1 tc2
trans ft1/1 ft2/2 ft4/3 ft5/4

state 1
props L.scc*19749 atm 12 plb*3 p2b*4 p4b*4 p5b*4 tc1 tc2
trans ft1/5 ft2/6 ft4/7 ft5/8 12f/9

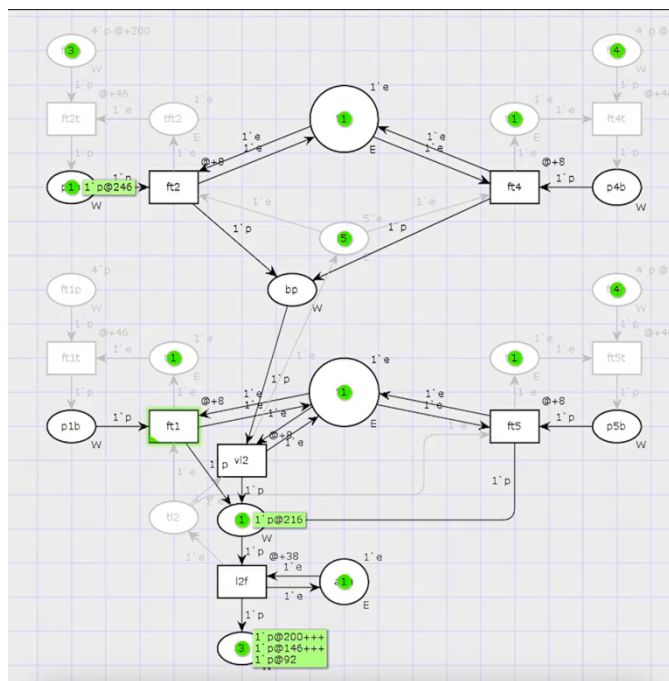
state 2
props L.scc*22799 atm bp plb*4 p2b*3 p4b*4 p5b*4 tc1 tc2
trans ft1/6 ft2/10 ft4/11 ft5/12 v12/13

state 3
props L.scc*23109 atm bp plb*4 p2b*4 p4b*3 p5b*4 tc1 tc2
trans ft1/7 ft2/11 ft4/14 ft5/15 v12/16

state 4

```

Figure 5.8 Analysis of final transient process Petri net in TINA.



Video 5.7 Clip of video for final transient process simulation using CPN Tools available at <https://www.youtube.com/watch?v=xnXOXkpKfcg>.



## CHAPTER 6

### CONCLUSION

The scheduling component of a semiconductor cluster tool is an expansive endeavor that requires highly experienced developers and engineers and a task that is growing in complexity every day. It is also the least researched area in the semiconductor industry. [4] As such, there could be undiscovered optimizations and improvements that might exist that have not been researched. Petri nets are one of the areas that could use more investment.

So far, Petri net scheduling of the cluster tool works well as long as the configuration of the cluster tool is static. However, that is not the reality of the semiconductor industry. The configurations of cluster tools are constantly changing with the desire for different behavior. That would require constant redevelopment of a new Petri net model, which is not trivial, without the ability to fully leverage the existing models. As the Petri net model increases in size, the computational complexity also increases exponentially, which is not tenable in the long term as the complexity of the cluster tools increases.

Modeling production cluster tools with real-world non-elegant requirements take away the elegance of existing theoretical solutions. Another area that requires exploration is the efficacy of switching between Petri net models for scheduling a heuristic rule-based approach. The relative computational and development cost should be researched.

On the other hand, Petri net models can be a great tool to deploy along with the heuristic rule-based approach at the launch of a new tool or configuration, as the model can explore the states that might or might not be encountered by developers as part of the testing and trial-and-error process. The Petri net model can catch the deadlocks, and other

undesirable states, mathematically. A Petri net model can be a foundation from which the developers' experience and creativity can evolve from, shortening the development time and allowing for the delivery of a more mature ruleset to market at launch.

## APPENDIX A

### GRAPHS & PROGRAM

Based on Dr. Zhao's insight into Petri net's ability to see over the horizon to make decisions on scheduling, the reachability graph was explored to calculate strongly connected components using Tarjan's algorithm, nodes of n-walks, and the shortest path using Dijkstra's algorithm. Google Colab for Python notebook was used, and the Network-X package was leveraged. The work is shared at:

<https://drive.google.com/drive/folders/18KzRtacbeJT1OnDbhKEW1Mq54SHGN6Lj>.

The motivation of the Python notebook is an exercise to take the output files from TINA and create multi-directed graphs, which we can use to get strongly connected components, shortest paths between states, and states of n-walk. We also wanted to filter the results using the number of tokens at different places to identify good or bad states. In general, any code cell is dependent on the prior code cells. Each code cell can be run independently, assuming all dependent cells have been run before.

The notebook has nine sections:

1. Overview
2. Libraries
3. File Import
4. Parsing Input
5. Make Graph
6. States Filter
7. Strongly Connected Components
8. N-Walks from a Node

## 9. Shortest Path(s)

The input file is imported as a dataframe from Google Drive by mounting Google Drive. If running this notebook locally, please comment out the Google Drive mount code and change that path to the appropriate input file that you would like to use. The input file is the output of the TINA Marking Graph (option-R). (Note: Please make sure to scroll all the way to the end of the TINA output prior to saving to ensure the full text is saved from TINA.)

For parsing, the input text file that is generated has three lines for each state; the first line gives us the state index. The second line gives us the different tokens for each marking/state. If the marking place has no tokens, it is not listed. If it has a single token, the name of the place is listed. If it has more than one token, the place name has a suffix \*, which is followed by an integer. The third line gives us the transitions. Each available transition has the transition with a suffix "/", followed by the target state.

This section has five parts:

- i. `parse_props` function: this function takes a list in and returns a dictionary
- ii. `parse_file` function: this function parses the input file and returns a tuple of `node_list` (tuple of nodes and labels), `place_dict` (dictionary for markings with places and tokens), and `dead_list` (list of dead states).
- iii. formatting `node_list` for a more human-readable format. The code is commented out, and its use is optional.
- iv. `dict_to_df` function: converts the dictionary to a dataframe, then replaces the NaNs with 0s and typecasts it into int

- v. this part prints out the dead state list. The code is commented out, and its use is optional. Only applicable when there are dead states in the input file.

NetworkX library is used to make a multi-directed graph, G, of the reachability graph using the `node_list` from the previous section (`parse_file` function). There are three optional code cells:

- i. prints the number of nodes and edges
- ii. draws the graph (Due to the size of the graph, it is not very usable.)
- iii. exports the graph into a `.gexf` format which can be read by the Gephi application to draw graphs.

States Filter is an optional section that takes the `props_df` generated by `diet_to_df` from the parsing input section and filters using the number of tokens in places. The first two code cells give us basic information about the dataframe to edit the filter.

For Strongly Connected Components, the program takes the multi-directed graph, G, from the NetworkX section and gets basic information about the G graph in terms of SCCs: how many there are, the length of each, and the largest SCC in the G graph using Tarjan's algorithm.

Based on the information gathered above, the `n-walk` function can be used to get the dataframe of nodes and their associated state information of states that are a certain distance from the source node, given a graph.

Based on the information gathered above, get the list of all available shortest paths, using Dijkstra's algorithm, from a source state to a target state.

Memory usage was measured with both psutil and tracemalloc. psutil measures the memory usage by the Python interpreter, and tracemalloc measures the Python dynamic memory allocation on the heap.

In Figure A.1, we can see both psutil and tracemalloc outputs for each function. For each function, we have a psutil that gives us the line-by-line memory usage with increments and occurrences. This is followed by current and peak memory usage by tracemalloc. For example, the function `run_nx`, which uses NetworkX to add edges and their corresponding labels, has incremental memory usage of (43.3 + 774.7 MiB) as per psutil, while tracemalloc shows that the function uses 648.25 MiB at peak.

The `get_sec` function, which gets the number of strongly connected components (SCC), their length(s), and printing the largest SCC, use 63.23 MiB at peak, according to tracemalloc, with only 0.6 MiB use according to psutil. The `n-walk` and `shortest path` functions also use 63.23 and 0.03 MiB, respectively, at peak.

```

Line # Mem usage Increment Occurrences Line Contents
=====
83 166.9 MIB 166.9 MIB 1 @profile
84 # code for which memory has to be monitored
85 # -----
86 # parsing and making lists out of the input file
87 # node_list, dead_list and place_dict are data structures/storage that we are using for
88 # this function. This function returns all three as a tuple.
89 def parse_file(data):
90     node_list = [] # list of tuples of nodes and labels
91     place_dict = {} # dictionary for places
92     dead_list = [] # list of dead states
93     # get the length/rows of the input text file since we are using pandas dataframe
94     length = data.shape[0]
95     # setting the starting row for processing
96     start_trans = 2
97     start_props = 1
98     # going through every state
99     for state in range(length // 3):
100
101         # this is for trans
102         # creates a list that is tokenized by space from the trans line in input file
103         trans = data[start_trans][6:].split(' ')
104         # iterates through the list
105         for i in range(len(trans)):
106             # index of the '/'
107             index = trans[i].find('/')
108             # gets the target state of the transition
109             transition = trans[i][index+1:]
110             label = trans[i][0:index]
111             # deadstate error handling
112             if len(transition) > 0:
113                 # adding things to the list as a tuple
114                 trans[i] = int(transition), label
115             else:
116                 dead_list.append(state) # for identifying dead states
117         # appending trans list to node_list
118         node_list.append(trans)
119         # iterating to the next state
120         start_trans += 3
121
122         # this is for props
123         # creating a list by tokenizing space
124         props = data[start_props][6:].split(' ')
125         place_dict[state] = parse_props(props)
126         start_props += 3
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

Figure A.1 Petri net graph memory usage

## APPENDIX B

### TINA INSTRUCTION

Link to TINA manual <https://projects.laas.fr/tina/manuals/tina.html>

During my work, the most used function in TINA is building the reachability graph (marking graph). This can be found at Tools> State Space Analysis> -R (with the liveness analysis option checked). If we set the output to default kts (.ktz), it generates a text file in a new window. It lists the number of places and transitions. Then it gives us the analysis of boundedness, liveness, and reversibility. This is then followed by the states and transitions information. First is the count of states and transitions. This is then followed by props, which are the elements of the states and transitions. It is then followed by psets, which is the flow information. Then we get the number of live and dead states and transitions. The text file that is generated has three lines for each state. The first line gives us the state index. The second line gives us the different tokens for each marking/state. If the marking place has no tokens, it is not listed. If it has a single token, the name of the place is listed. If it has more than one token, the place name has a suffix ‘\*’, which is followed by an integer. The third line gives us the transitions. Each available transition has the transition with a suffix ‘/’, followed by the index of the target state.

Note: We can save the file as a text file. However, to get the full list, we have to scroll all the way to the end of the document prior to saving it. Otherwise, only a partial list will be saved as a .ktz file. However, you can rename the file as .txt during or after saving the file.



## APPENDIX C

### CPN TOOLS INSTRUCTION

Link to CPN Tools documentation <http://cpntools.org/2018/01/16/documentation-2/>

For this exploration purpose, CPN Tools was used as a visual simulator of wafer movement. The simulation files are shared at:

<https://drive.google.com/drive/folders/18KzRtacbeJT1OnDbhKEW1Mq54SHGN6Lj>.

.cpn file that simulates the wafer flow of batch-begin, batch-end, and steady-state.

The simulation is not a circular process. It will only process the number of wafers specified in the cu place. To change the number of wafers in cu, you can click on the value to the top right of the cu place, which is usually an integer followed by ‘u’, and change the integer value to the number of the wafer you would like to simulate. You can also click on the cu place and press tab to cycle through the parameters of it and change the integer value.

To simulate, please drag the word Simulation from the menu on the left-hand side onto the darker blue area of the application on the right. It will open the Simulation menu. Please press the play button to simulate. The play button has a number below it that specify how many steps it will execute. You can change it by right-clicking on the play button to Set Options for the play button. You can use the rest of the control buttons to control the simulation, including simulating one step at a time.

## REFERENCES

- [1] B. Berthomieu, F. Vernadat, and S. dal Zilio, "TINA toolbox - Time petri Net Analyzer - by LAAS/CNRS," *TINA Toolbox*.  
<https://projects.laas.fr/tina/index.php> (accessed Jul. 12, 2022).
- [2] CPN Tools, "Anti places/limit places – CPN Tools," *CPN Tools*, Jan. 11, 2018.  
<https://cpntools.org/2018/01/11/anti-places-limit-places/> (accessed Jun. 11, 2022).
- [3] CPN Tools, "CPN Tools – A tool for editing, simulating, and analyzing Colored Petri nets," *CPN Tools*. <https://cpntools.org> (accessed Dec. 13, 2022).
- [4] J. W. Fowler, "Scheduling Problems in Semiconductor Wafer Fabrication Facilities: Part 1," <https://schedulingseminar.com/>, Mar. 03, 2022.  
<https://www.youtube.com/watch?v=5ZCtbU1VR3s> (accessed Mar. 04, 2022).
- [5] B. Hruz and M. Zhou, *Modeling and Control of Discrete-event Dynamic Systems*. Springer Science & Business Media, 2007.
- [6] O. Kilincci, "A Petri net-based heuristic for simple assembly line balancing problem of type 2," *The International Journal of Advanced Manufacturing Technology*, vol. 46, no. 1–4, pp. 329–338, Jan. 2010, doi: 10.1007/s00170-009-2082-z.
- [7] C. A. Petri, "Kommunikation mit Automaten," Dissertation, Universität Hamburg, 1962.
- [8] C. Petri and W. Reisig, "Petri net," *Scholarpedia*, vol. 3, no. 4, p. 6477, 2008, doi: 10.4249/scholarpedia.6477.
- [9] Y. Qiao, M. Zhou, N. Wu, Z. Li, and Q. Zhu, "Closing-Down Optimization for Single-Arm Cluster Tools Subject to Wafer Residency Time Constraints," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 11, pp. 6792–6807, Nov. 2021, doi: 10.1109/tsmc.2020.2964032.
- [10] J. Wang, H. Hu, C. Pan, Y. Zhou, and L. Li, "Scheduling dual-arm cluster tools with multiple wafer types and residency time constraints," *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 3, pp. 776–789, May 2020, doi: 10.1109/jas.2020.1003150.
- [11] S. Wang *et al.*, "Computation of an emptiable minimal siphon in a subclass of Petri nets using mixed-integer programming," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 1, pp. 219–226, Jan. 2021, doi: 10.1109/jas.2020.1003210.
- [12] Wikipedia Contributors, "Petri net," *Wikipedia*, Nov. 08, 2019.  
[https://en.wikipedia.org/wiki/Petri\\_net](https://en.wikipedia.org/wiki/Petri_net) (accessed Nov. 18, 2021).

- [13] N. Q. Wu and M. Zhou, "A Closed-Form Solution for Schedulability and Optimal Scheduling of Dual-Arm Cluster Tools With Wafer Residency Time Constraint Based on Steady Schedule Analysis," *IEEE Transactions on Automation Science and Engineering*, vol. 7, no. 2, pp. 303–315, Apr. 2010, doi: 10.1109/tase.2008.2008633.
- [14] H. H. Xiong and M. Zhou, "Scheduling of semiconductor test facility via Petri nets and hybrid heuristic search," *IEEE Transactions on Semiconductor Manufacturing*, vol. 11, no. 3, pp. 384–393, Aug. 1998, doi: 10.1109/66.705373.
- [15] F. Yang, N. Wu, Y. Qiao, M. Zhou, and Z. Li, "Scheduling of Single-Arm Cluster Tools for an Atomic Layer Deposition Process With Residency Time Constraints," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 3, pp. 502–516, Mar. 2017, doi: 10.1109/tsmc.2015.2507140.
- [16] Z. Zhao, S. Liu, M. Zhou, D. You, and X. Guo, "Heuristic Scheduling of Batch Production Processes Based on Petri Nets and Iterated Greedy Algorithms," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 1, pp. 251–261, Jan. 2022, doi: 10.1109/tase.2020.3027532.
- [17] M. Zhou and K. Venkatesh, *Modeling, Simulation, and Control of Flexible Manufacturing Systems : A Petri Net Approach*. Singapore: World Scientific, 1999.
- [18] M. Zhou and N. Wu, *System Modeling and Control with Resource-Oriented Petri Nets*. New York: CRC Press, 2017.
- [19] Q. Zhu, M. Zhou, Y. Qiao, and N. Wu, "Petri Net Modeling and Scheduling of a Close-Down Process for Time-Constrained Single-Arm Cluster Tools," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 3, pp. 389–400, Mar. 2018, doi: 10.1109/tsmc.2016.2598303.
- [20] Q. Zhu, Y. Qiao, N. Wu, and Y. Hou, "Post-processing time-aware optimal scheduling of single robotic cluster tools," *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 2, pp. 597–605, Mar. 2020, doi: 10.1109/jas.2020.1003069.
- [21] R. Zurawski and M. Zhou, "Petri Nets and Industrial Applications: A Tutorial," *IEEE Transactions on Industrial Electronics*, vol. 41, no. 6, pp. 567–583, 1994, doi: 10.1109/41.334574.