Department of Social Systems and Management

Discussion Paper Series

No.1191

# Managing Assembly Production Processes with KANBAN and CONWIP

by

Ryo Sato, Yaghoub Khojasteh-Ghamari

February 2008

UNIVERSITY OF TSUKUBA
Tsukuba, Ibaraki 305-8573
JAPAN

# Managing Assembly Production Processes with KANBAN and CONWIP

Ryo Sato[†], Yaghoub Khojasteh Ghamari[‡]

[†]*Department of Social Systems and Management, University of Tsukuba, Japan*
[‡]*Graduate School of Systems and Information Engineering, University of Tsukuba, Japan*

**Abstract**
To control the production of parts in a production process, managers can choose a proper production control policy. This paper provides a performance analysis of production control systems in a business process, which is modeled as a discrete-event system and depicted by AID (Activity Interaction Diagram). KANBAN and CONWIP controls are focused on and analyzed. The periodic behavior of a token transaction system and the concepts of critical circuit and tokens are used. When a business process behaves periodically, Little's law can be used to calculate the cycle time, inventory, and throughput of the system. By using the theory of token transaction systems, and employing the law, we show how minimum WIP (Work-In-Process) of a system can be calculated that allows the system to have maximum possible throughput. As an application of the theory, we provide a performance comparison between KANBAN and CONWIP. The results show that there is no general superiority between KANBAN and CONWIP. Appropriate design of the whole system decides which one is superior in certain situations.

*Keywords:* Production control systems; KANBAN; CONWIP; token transaction systems; Little's law; critical circuit

## 1. Introduction

In order to synchronize production and sales delivery, production processes need to be controlled. KANBAN and CONWIP control mechanisms are successful examples of card-based production control systems. (In the following, we simply write KANBAN and CONWIP to mean respective KANBAN and CONWIP controlled production processes, as long as it is clear from the context.) Since the references for KANBAN and CONWIP are many, we just put Monden (1998) for KANBAN, and Spearman *et al.* (1990) and Hopp and Spearman (2001) for CONWIP. In the KANBAN, information is sent from a station only to its immediate proceeding station, while in CONWIP (CONstant Work-In-Process), information about a product demand flows directly from the final buffer to the first station. Since they have different mechanisms, a number of comparative studies have been conducted. According to the survey by Framinan *et al.* (2003), in comparison of the two, many authors insist that CONWIP outperforms KANBAN when processing times on component operations in production processes are variable. Gstettner and Kuhn (1996) showed, however, KANBAN achieves a given throughput with less work-in-process (WIP, for short) at finished part buffer.

All of the papers, which Framinan *et al.* (2003) cited for comparison of the two policies in optimal performance, had focused on serial production line or enhanced serial production line for their purpose of research. Among them, Bonvik *et al.* (1997), Bonvik and Gershwin (1996), Paternina-Arboleda and Das (2001), and Yang (2000) had used simulation for analysis. Spearman and Zazanis (1992) and Muckstadt and Tayur (1995) had shown analytical result on card-based control for serial production processes. Processing times for operations in a production process are varying with respective exponential distributions. When the same number of cards is used in both CONWIP and KANBAN, Spearman and Zazanis (1992) have shown that the throughput of KANBAN does not exceed that of CONWIP. They pointed out that it holds true because circuits in CONWIP are virtually divided into smaller circuits in KANBAN, and then the cards in KANBAN tend to be "blocked". Muckstadt and Tayur (1995) had used a generalized serial production line in analyzing card-based production control systems. Series of machines form a cell, and series of cell are connected as a production line. In a cell, CONWIP control is used. If each cell has only one machine, the whole system is virtually a KANBAN system. In a generalized production line, four sources of variability are considered. They are processing time variability, machine breakdowns, rework and yield loss. It has been shown that if we deploy more cards then the average waiting time of production orders could decrease or remain equal.

As Framinan *et al.* (2003) pointed out, when two control policies are compared, both should be optimally tuned. Otherwise, we cannot say the amount of average WIP, for example, is less or more. Usual manufactured products have BOMs and corresponding routings. Takahashi *et al.* (2005) compared CONWIP and KANBAN for tree-shaped production process.

This paper proposes a novel design discipline for card-based control of production process, by developing the theory of token transaction systems. The theory shows how the three indices represented in Little's law (Little, 1961) are decided by the structure of a production process with control-cards and deployment of WIP. That is, the relation of WIP, cycle time and throughput on specific sub-network of production process is clarified. In other word, we show how the Little's law should be used in the design of card-based production control systems. As an application of the theory, we resolve complicated result of comparison between CONWIP and KANBAN. In doing so, this theory does not restrict the target of analysis to serial production processes, but any shaped processes can be virtually analyzed.

The rest of the paper is organized as follows. In Section 2, the concept of token transaction system and related definitions are introduced. Section 3 provides the properties of token transaction systems that are used in analysis. In Section 4, CONWIP and KANBAN are analyzed so that we can clearly understand why the comparisons of CONWIP and KANBAN became complicated. Section 5 is the conclusion.

## 2. Modeling production process

In modeling production processes with control mechanisms, this paper employs the concept of business transaction system (Sato and Praehofer, 1997) that is based on the DEVS formalism for discrete-event systems (Zeigler, 1976). In general, dynamic behavior of a discrete-event system requires causality. According to Mesarovic and Takahara (1975), a dynamic system has a state transition function if and only if the system is causal. Sato (2001) showed that a DEVS model always brings corresponding state transition function and it is unique up to isomorphism. In this sense, the DEVS formalism is universal. Thus, adopting the DEVS formalism is fairly common decision in modeling discrete-event systems. In a business transaction system, the components and connecting structure are represented by activity interaction diagrams (AID, for short).

**Definition 1**. Activity Interaction Diagram (AID) (Sato and Praehofer, 1997)
An activity interaction diagram is a diagram that has three kinds of components. They are activities, queues, and connecting arrows. Activities should be connected with queues, and vice versa. That is, in the graph theoretic sense, an AID is a directed bipartite graph.

In this paper we consider specific type of business transaction systems, where queues are simplified as usual FIFO (first-in, first-out) queues to store objects called *tokens* and every queue can have at most one input and output arrow. We call such system a *token transaction system*. In a token transaction system, tokens represent parts, products, actors, or data. Queues are also referred as connecting queues. The AID of a token transaction system for a simple assembly process is depicted in Figure 1, where activities and queues are represented by squares and ovals, respectively. It shows a serial production process governed by CONWIP. The purchased material, $m$, is processed by operations $p_1$ through $p_4$ to be a product which is stored in the place $b$. Each of those operations $p_1$, $p_2$ and $p_3$ produces its output part which will be stored in $b_1$, $b_2$, and $b_3$, respectively. The workers for operations are represented by tokens in $w_i$ ($i$ = 1, 2, 3, 4). The queue $C$ represents the storage place of cards. A card functions as a production order for $p_1$. Let $A$ be the set of internal activities, and $Q$ the set of queues. The number of tokens in a queue $C$ is denoted by $|C|$. The output queues of an activity are specified as one of two types. An output queue of a type gets one token from the activity when it starts, while the other type queue gets a token when the activity finishes. The former queues are called ones of $Q_S$ type, and the others are $Q_F$ type. An activity can have both types output.

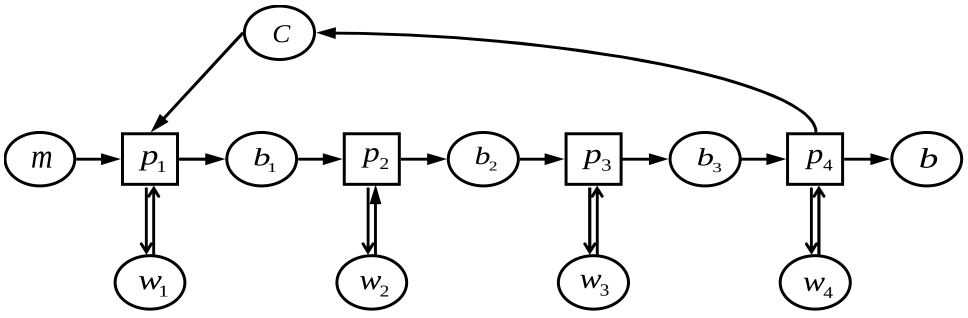


Figure 1. A serial line with CONWIP

Now we define the dynamics of a token transaction system. The time evolution of a token transaction system is defined by the state transition function, which is originally defined by the set theoretic notation by Sato and Praehofer (1997). For a token transaction system we can use state transition table, because the content of queue variables are simple FIFO tokens.

*Rule of transfer of tokens in a token transition system:*

An activity starts when its starting condition is met. That condition is defined by relation between the input queues. Once started, an activity will finish after prescribed processing time (or holding time) for a token. When an activity starts, one token is removed from each input of the activity, one token is held in the activity during the processing time, and one token is added in the outputs of the $Q_S$ type. When an activity finishes, one token is added to each of the output queues of $Q_F$ type of the activity.

With the above rule, the state transition of a token transaction system is defined as shown in Figure 2, and then brings us the state transition table, Table 1. In Figure 2, an activity is said to be "imminent" if its holding time had elapsed from its starting time. There might be several activities which are imminent at a time. When imminent activities finish, the output queues of $Q_F$ type of each imminent activity get respective tokens. When an activity can start, it must start. If no activity can start, then the placement of tokens in the whole process remains the same until the next event comes. The time instant of the next event is defined as the minimum of the due times of activities in operation. So, the next event will become the next "current time" in the state transition table, and then it continues.

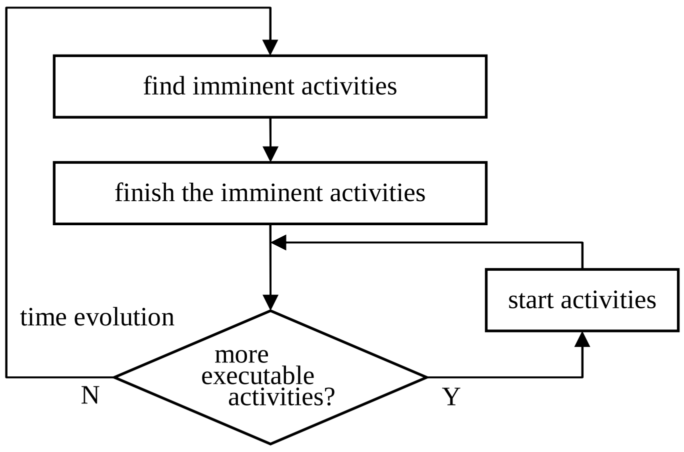


Figure 2. Flow chart of dynamic behavior of a token transaction system

Table 1. State transition table of a CONWIP (Initial condition: *C* has 4 tokens. Numbers of actors for $p_1$, $p_2$, and $p_4$ are 1, respectively. Number of actors for $p_3$ is two. Each activity is idle and inventory in each of $b_1$, $b_2$, $b_3$ and $b$ is 0.)

| time | $C$ | $p_1$ | $w_1$ | $b_1$ | $p_2$ | $w_2$ | $b_2$ | $p_3$ | $w_3$ | $b_3$ | $p_4$ | $w_4$ | $b$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 22 | 0 | 1(2) | 0 | 0 | ---- | 1 | 1 | 1(7),1(10) | 0 | 0 | 1(5) | 0 | 1 |
| 24 | 0 | ---- | 1 | 0 | 1(3) | 0 | 1 | 1(5),1(8) | 0 | 0 | 1(3) | 0 | 1 |
| 27 | 0 | ---- | 1 | 0 | ---- | 1 | 2 | 1(2),1(5) | 0 | 0 | ---- | 1 | 2 |
| 29 | 0 | 1(2) | 0 | 0 | ---- | 1 | 1 | 1(3),1(12) | 0 | 0 | 1(5) | 0 | 2 |
| 31 | 0 | ---- | 1 | 0 | 1(3) | 0 | 1 | 1(1),1(10) | 0 | 0 | 1(3) | 0 | 2 |
| 32 | 0 | ---- | 1 | 0 | 1(2) | 0 | 0 | 1(9),1(12) | 0 | 1 | 1(2) | 0 | 2 |
| 34 | 0 | 1(2) | 0 | 0 | ---- | 1 | 1 | 1(7),1(10) | 0 | 0 | 1(5) | 0 | 3 |

The time evolution of CONWIP system in Figure 1 is determined by specifying the starting condition of activities and movement of tokens. By assuming that we always have enough material, we do not have to care about $m$. So, $p_1$ starts its processing if more than one card exists in $C$ and if the worker is available (that is, if the worker is not busy). When $p_1$ finishes, it outputs a token to $b_1$. One token in $b_1$ represents combination of a part and a card. Each of the operations $p_2$, $p_3$, and $p_4$ will start its operation if more than one pair of part and an attached card exist in the respective input buffer, and if the respective actor is available. When $p_4$ starts, it also outputs a card to *C*. As a whole, Table 1 will come out. In

Table 1, "----" represents that there is no token being processed. That is, the corresponding worker is idle. "1(2)", for example, shows that one token is being processed and it will finish (be imminent) after 2 minutes. As like the $p_3$ column, two tokens can be processed each of which will be imminent independently.

We need to further investigate some properties of token transaction systems. We first define "never-stopping" transaction systems.

**Definition 2**. Live system (Murata, 1992)
A token transaction system is said to be *live* with respect to an initial state $s$, if the state transition table will last forever from the state $s$. If a token transaction system is live with respect to a state $s$, then it is said to be live.

Even if a transaction system is live, the system might have activities that will never start. In that case delete those activities with related connecting (input/output) arrows. Apparently, the resultant transaction system has the same dynamic behavior. Notice that the static structure of a token transaction system is modeled by an AID. A *path* is a series of activities and queues that follows the direction of connecting arrows among them. If the start and end of a path are the same activity or queue, then the path is called a *circuit*. If a circuit contains different activities and queues (except the start and end), then it is called an *elementary circuit*. When a circuit contains $Q_S$ type queues, then the activities whose outputs are the queues can be eliminated to form the (shorter) circuit. For example, $p_1 b_1 p_2 b_2 p_3 b_3 p_4 C p_1$ is a circuit and $p_1 b_1 p_2 b_2 p_3 b_3 C p_1$ is also a circuit, because $C$ is a $Q_S$ type output queue of $p_4$. For a circuit $C$, the set of activities in $C$ is denoted by $A(C)$. The *cycle mean* of a circuit is defined as the sum of the holding time of the activities of the circuit, divided by the number of tokens in the circuit. The *maximum cycle mean*, $\lambda$, of an AID is the maximum value of all cycle means (Baccelli *et al.*, 1992) and is given by

$$\lambda = \max_{\zeta} \frac{|\zeta|_h}{|\zeta|_t},$$

where, $\zeta$ ranges over the set of elementary circuits of the AID, and $|\zeta|_h$ denotes the sum of the holding times of the activities in the circuit, and $|\zeta|_t$ denotes the number of tokens in the circuit. It is clear that any non-elementary circuit has the cycle mean which is less than or equal to the maximum cycle mean. All the circuits that have maximum value of cycle mean are called *critical circuits*.

**Definition 3**. Strong connectivity of AID（Sato and Kawai, 2007）
Consider an AID of a token transaction system. Let $A$ and $Q = Q_S \cup Q_F$ be the sets of

activities and queues, respectively. Let $a \in A$ and $q \in Q$ be arbitrary. If there exist a path from $a$ to $q$ and one from $q$ to $a$, then the AID and the token transaction system are said to be strongly connected.

## 3. Design discipline of token transaction systems

The Little's law governs dynamics of business process, when the process is in steady state. Periodic behavior is a kind of steady state. The law says rigorous relation among cycle time, WIP, and throughput.

### 3.1. *WIP, TH and CT in cyclic behavior*

Average WIP is defined as average value of inventories. If we assume the inventory of a trading good moves out in constant pace and the good is replenished once in every time interval $N$, then its inventory trajectory is depicted by Figure 3. Let us denote the inventory level at time $t$ as $w(t)$. Then, the average WIP is calculated as follows.

$$
\begin{aligned}
WIP &= \lim_{T\to\infty}\frac{1}{T}\int_0^T w(t)dt \\
&= \frac{1}{N}\int_0^N \left(-\frac{V}{N}t + V\right)dt = \frac{V}{N}\left[-\frac{t^2}{2N}+t\right]_0^N = \frac{V}{2}
\end{aligned}
$$

Thus, it suffices for calculation of average value to consider a period, instead of infinite interval. Similarly, average throughput (TH) and cycle time (CT) can be calculated for a period. WIP is usually represented as sum of safety and cycle stocks, where the former is considered as buffer for randomness. Since this paper focuses on deterministic model, WIP contains only cycle stock.

In the following, if it is clear from the context, we simply write *WIP* to mean *average WIP*. And, we use TU to mean "time units" and PC to mean "pieces". TU can be interpreted as week, hour, minute, and so on.

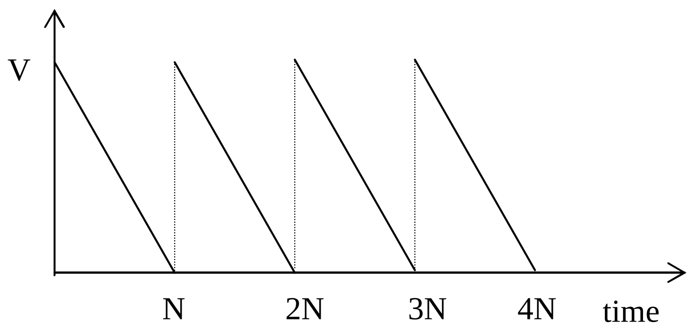


Figure 3. Cycle inventory

## 3.2. *Little's law in periodic behavior*

In this section, dynamic properties are investigated.

**Proposition 1** (Sato and Kawai, 2007). For a circuit of a token transaction system, the number of tokens in the circuit remains the same at any state transition.

The proof of above proposition is based on the one for an event graph, which is given by Murata (1992).

In order to be live for a strongly connected token transaction system, it suffices that every circuit has tokens, accordingly. Especially, if a strongly connected system has tokens in every circuit at initial state, then it is live. Furthermore, since we define WIP by the average number of tokens, WIP remains the same in a strongly connected system.

**Proposition 2** (Sato and Kawai, 2007). In the state transition table of a token transaction system that is both strongly connected and live, the number of possible values of remained time for an activity is finite.

**Proposition 3** (Sato and Kawai, 2007). A token transaction system that is strongly connected and live has periodic behavior.

Since the holding times of activities are not integer but real, the above propositions are not trivial.

**Proposition 4** (Sato and Kawai, 2007). Consider a token transaction system that is strongly connected and live, and assume that it is in the periodic behavior. Then, every activity has the same number of commencement in the period.

The number of commencement in a period is called the *activation frequency* of the system. Notice that the numbers of commencement and finish of an activity in a period are the same so that the definition is well defined. The throughput of a strongly connected and live token transaction system is defined as average value of the number of output tokens from an activity of the system. Since the activation frequency is the same for all of the activities in the system, this definition of throughput is well defined. The cycle time of a circuit is defined as the elapsed time for a token to go round on the circuit in the periodic behavior.

**Theorem 1.** Consider a strongly connected and live token transaction system. Let $TH$ be the throughput of the system in the periodic behavior, $C$ a circuit of the system, $w_C$ the average WIP of tokens on $C$, and $CT_C$ the cycle time of $C$. Then the Little's law holds on

$C$. That is, $CT_C = \dfrac{w_C}{TH}$ holds true.

The above proposition shows that the Little's law holds only for circuits. In other words, the average number of total inventories of a production system does not work as the WIP term in the law.

Since the maximum cycle mean is determined by the structure of the system and the placement of WIP, the following two propositions show how to design the throughput of a token transaction system by specifying the structure of and placement of WIP in the system.

**Proposition 5.** Consider a token transaction system that is strongly connected and live. Assume that it is in the periodic behavior. Then, any activity of a critical circuit of the system never be blocked its commencement. That is, if an activity on a critical circuit is not busy, and if the activity's input queue on the circuit gets any token, then it starts its processing immediately.

**Proposition 6.** Consider a strongly connected and live token transaction system. Let $\lambda$ be its maximum cycle mean, and $TH$ the throughput. Then, $TH = \dfrac{1}{\lambda}$.

In order to increase the maximum throughput of the system, the maximum cycle mean should be decreased. It means that the structure or WIP placement should be changed. If either factor changes, then another circuit can become critical. This makes situation complicated so that every circuit should be considered and that focusing on the current critical circuit is not enough to improve the performance of a production system.

The throughput of a token transaction system can be designed as follows. Figure 4(a) is the strongly connected and live token transaction system under concern. In order to design the best throughput of the system, we can delete the input and output so that the resultant system is strongly connected. As like Figure 4(b), the input and output systems can be attached, according to the importance of and/or interest in activities and the corresponding queuing variables. For example, the input system is a procurement division or supplier, and the output is a delivery division or outside wholesaler. Theory of business transaction system (Sato and Praehofer, 1997) assures that any composition of token transaction systems, where the connection of systems accords with AID structure is also a token transaction system as a whole.

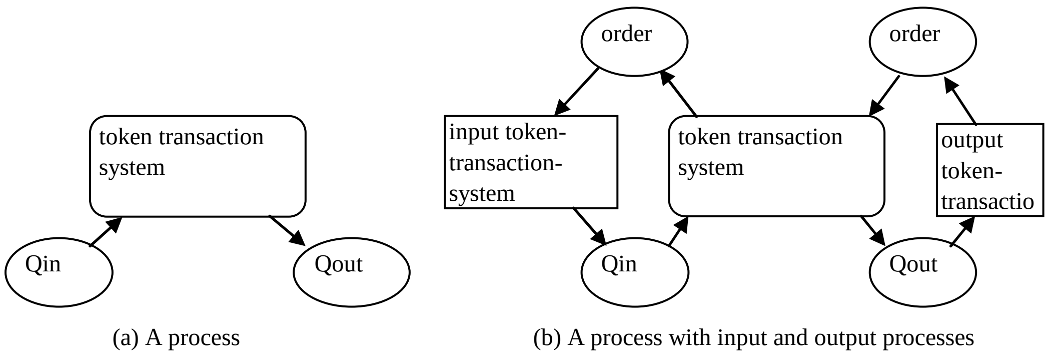


Figure 4. Designing throughput of a process

The system, of which input and output have the same throughput, provides us with a basic case of equilibrium behavior. The production control systems in Figure 4(a) and 4(b) have the following dynamic feature, by using Propositions 6 and 7.

(1) Let the throughput of the system in Figure 4(a) be $TH_p$, and that of the input and output in Figure 4(b) $TH$.
(2) If the input and output run slower than the system, that is, if $TH \le TH_p$ holds, then the whole system in Figure 4(b) shows equilibrium behavior with the throughput, $TH$. If $TH > TH_p$, then the whole system will never reach into equilibrium behavior, by accumulating increasing inventory at the input queue.
(3) Therefore, when $TH = TH_p$ holds, then the whole process attains the maximum throughput with respect to the input and output. In other words, $TH_p$ decides the best possible throughput of the whole system. That is, the system shows the same performance as we assume that the input provides infinite capacity and the output delivers finished part as soon as completion.
(4) Even if the whole system attains $TH_p$, the WIP is not necessarily minimum. Proposition 7 tells us how the minimum WIP in the whole system can be deployed.
(5) If the current best-possible throughput is insufficient, then the critical activities should be improved. In order to be non-critical for an activity, making the processing time shorter or increasing the numbers of actors and cards is effective. By powering up the critical activity, another activity then becomes critical.

The sum of WIPs in a KANBAN or CONWIP system is focused on sometimes. It has practical significance. The sum of WIPs is called the system WIP in this paper. As the following Propositions 7 and 8 show, the system WIP of a production process does not uniquely determine the critical circuits or the throughput.

**Proposition 7.** Consider a token transaction system that is strongly connected and live. Then, there exists the least system WIP that attains the throughput of the system.

## 4. Analysis of card-based production control systems

By applying the theory developed so far, we compare CONWIP and KANBAN for two different production processes. One type is serial line and the other is tree shape (or, a bill of materials plus a routing). In comparison of different control schemes, as Framinan *et al.* (2003) pointed out, optimized parameters should be used. The meaning of optimization of parameters in this paper refers to attainment of the maximum throughput of the whole system with a control schema. As the theory of this paper showed so far, maximum throughput of a token transaction system is determined by the structure WIP placement. As the optimized WIP, we need to consider minimum system WIP in the sense of Proposition 7.

One of the reasons of complicated aspect of the comparison (Framinan *et al.*, 2003) is as follows. Assume that a controlled production process is optimal. That is, it has the throughput with minimum WIP. Assume that one token is added. Then the system WIP certainly increases. But if the added token changes the critical circuit and throughput, this new system WIP could be still minimum with respect to the throughput. Also, deletion of a token could bring the change of critical circuit. In this way, finding the minimum amount of system WIP does not allow a "linear" search. So, we needed a theory.

Notice that tokens in a token transaction system correspond to cards, parts, or actors. Since tokens decide whether an activity can start processing, any of the three should be considered in analysis and design of dynamic behavior. Deployment of tokens in a system decides the throughput and WIP, and hence the cycle time. In the following, a circuit consists of an activity and its actors, such as $p_2 w_2 p_2$ is called an *activity circuit*.

### 4.1. *Serial production system*

Using analytical queuing network models, Gstettner and Kuhn (1996) provided a quantitative comparison between CONWIP and KANBAN with respect to WIP and throughput in a serial production line including six workstations with exponentially distributed processing times. According to their results, KANBAN can result in a lower average WIP level than CONWIP for a given production rate if the card distribution in the KANBAN is chosen appropriately. They defined the average number of finished parts in the output buffer of a station as the average WIP.

We present comparative analysis of the performance measures between CONWIP and KANBAN in a serial production process shown in Figures 1 and 5, respectively.

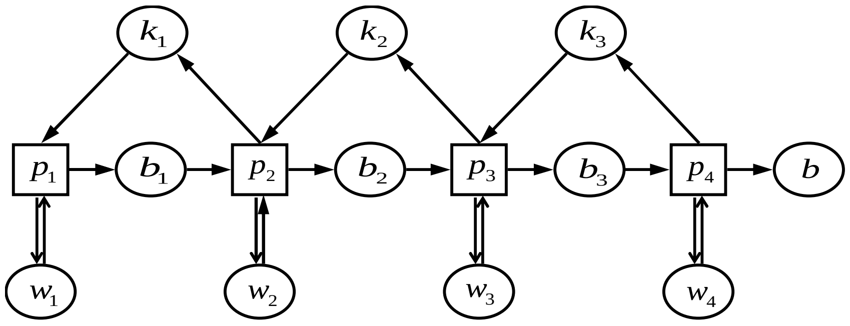


Figure 5. A serial line with Kanban system

The CONWIP process in Figure 1 is composed of four processes $p_1$ through $p_4$, and respective actors $w_1$ through $w_4$, and each process has output $b_i$ or $b$. The process $p_3$ has two actors, while each of the others has only one actor. The corresponding KANBAN process for the same serial production line is specified as Figure 5. The first process $p_1$ starts when more than one token is available in each of its inputs, $w_1$ and $k_1$. We assume that enough material $m$ is always available so that we do not take care of. When $p_1$ finishes, a token will be added into each of $b_1$ and $w_1$. The process $p_2$ starts when more than one token is available in each of its inputs $w_2$, $k_2$ and $b_1$. A token is produced in $k_1$ and $p_2$ when $p_2$ starts. The outputs of $p_2$ are $b_2$ and $w_2$. The process $p_3$ and $p_4$ works similarly. In the following, we show an example of a serial production line controlled by KANBAN, with respective state transition table (Table 2).

Case **Serial-KANBAN.** Table 2 gives the state transition table for the serial production line shown in Figure 5. Initial inventory for every part is set to 0, and initial cards are set as $k_1 = k_2 = 1$ and $k_3 = 2$. Each of $p_1$, $p_2$ and $p_4$ has one actor, while $p_3$ has 2. The system shows a periodic behavior every 12 [TU]. Each activity starts twice in a period. The throughput is 2/12, and the system WIP is equal to 6.17. It can be verified that the number of system WIP is minimum to attain the throughput 2/12. Later, in Tree-CONWIP-2 case, we will show how to calculate system WIP from state transition tables.

Case **Serial-CONWIP**. The state transition table for the same process under CONWIP has been given in Table 1. Four cards are initially assigned in the system. Initial inventories as well as the respective number of actors are the same as the case above. The period is 12 [TU], the throughput is 2/12, and the system WIP is 6.17, which is the minimum value to attain the throughput.

Table 2. State transition of Serial-KANBAN for a period

| time | $k_1$ | $p_1$ | $w_1$ | $b_1$ | $k_2$ | $p_2$ | $w_2$ | $b_2$ | $k_3$ | $p_3$ | $w_3$ | $b_3$ | $p_4$ | $w_4$ | $b$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 22 | 0 | 1(2) | 0 | 0 | 0 | 1(3) | 0 | 0 | 0 | 1(7), 1(12) | 0 | 0 | 1(5) | 0 | 1 |
| 24 | 0 | ---- | 1 | 1 | 0 | 1(1) | 0 | 0 | 0 | 1(5), 1(10) | 0 | 0 | 1(3) | 0 | 1 |
| 25 | 0 | ---- | 1 | 1 | 0 | ---- | 1 | 1 | 0 | 1(4) ,1(9) | 0 | 0 | 1(2) | 0 | 1 |
| 27 | 0 | ---- | 1 | 1 | 0 | ---- | 1 | 1 | 0 | 1(2), 1(7) | 0 | 0 | ---- | 1 | 2 |
| 29 | 0 | 1(2) | 0 | 0 | 0 | 1(3) | 0 | 0 | 0 | 1(12), 1(5) | 0 | 0 | 1(5) | 0 | 2 |
| 31 | 0 | ---- | 1 | 1 | 0 | 1(1) | 0 | 0 | 0 | 1(10), 1(3) | 0 | 0 | 1(3) | 0 | 2 |
| 32 | 0 | ---- | 1 | 1 | 0 | ---- | 1 | 1 | 0 | 1(9), 1(2) | 0 | 0 | 1(2) | 0 | 2 |
| 34 | 0 | 1(2) | 0 | 0 | 0 | 1(3) | 0 | 0 | 0 | 1(7), 1(12) | 0 | 0 | 1(5) | 0 | 3 |

In both cases, Serial-CONWIP and Serial-KANBAN, the optimum system WIPs to attain the same level of throughput are the same. The following proposition claims that this statement holds true when the same total number of cards is employed in the both systems.

**Proposition 8.** Consider the serial production process shown in Figures 1 and 5 with CONWIP and KANBAN, respectively. Assume that both systems have the same actors for respective processes, the same activation frequency, and the same throughput. Let *N* and *K* be the total number of cards in CONWIP and KANBAN, respectively. Then, we have the following.

(i) $N < K$ if and only if $W_C < W_K$,

(ii) $N = K$ if and only if $W_C = W_K$,

where $W_C$ and $W_K$ are the average system WIP for CONWIP and KANBAN, respectively.

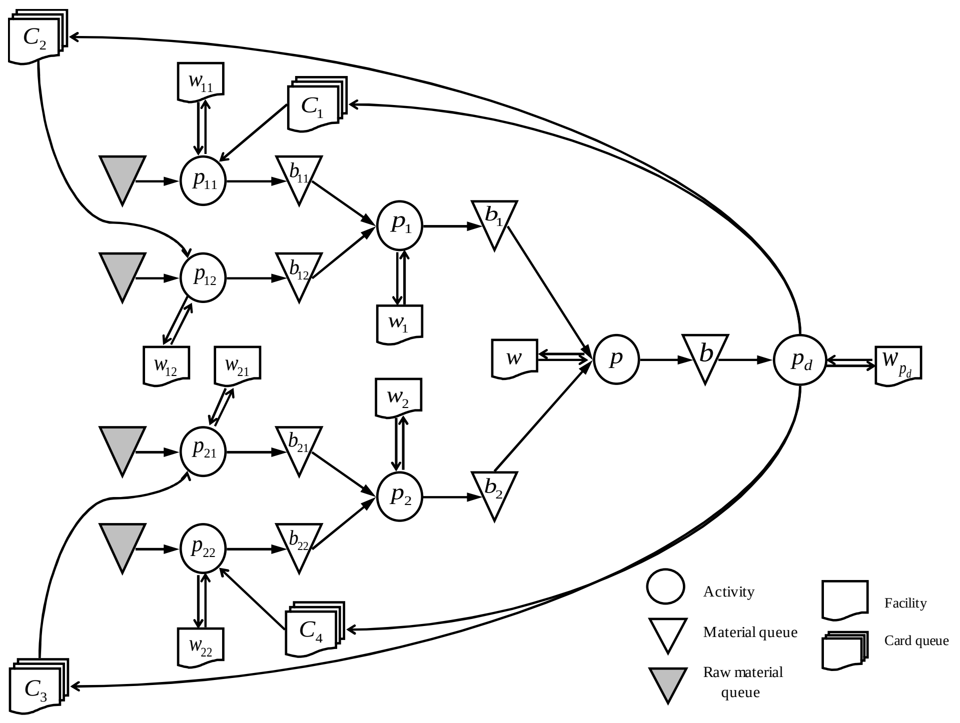
This proposition resolves the complicated situation on a serial production line. In this proposition, the definition of WIP is different from that of Gstettner and Kuhn (1996). The system WIP is a factor that determines throughput. Therefore, if we focus on the average value of the final product without considering the other inventories in the system, then that amount alone does not bring us useful information. And it is not expected that any kind of optimality can be attained with respect to the three dynamic indices used in the Little's law.

## 4.2. *Tree-shaped production process*

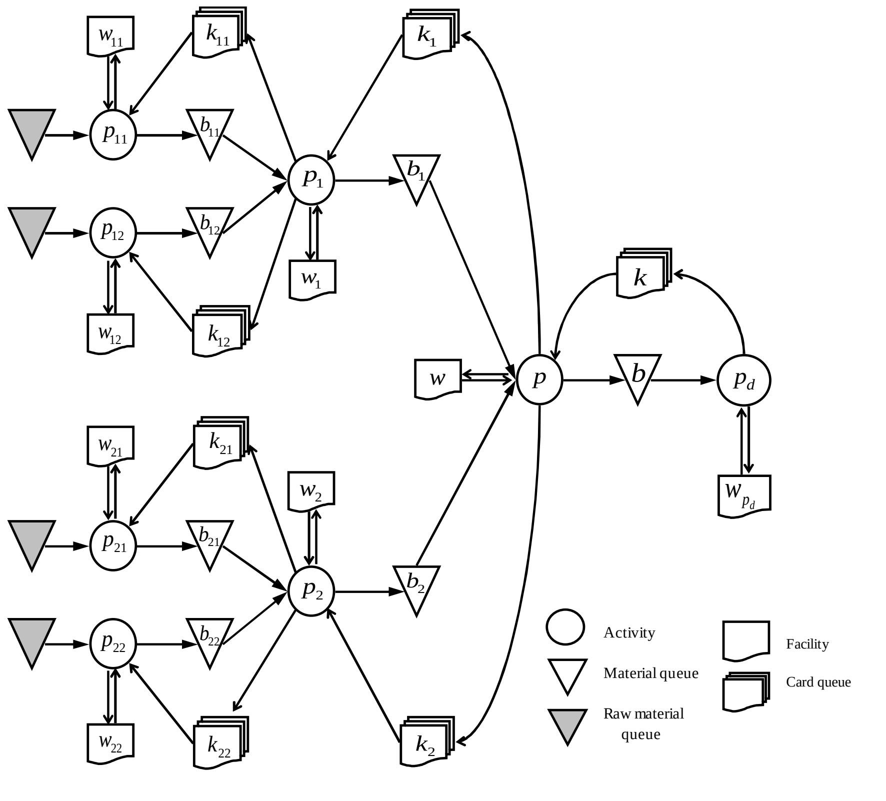
Takahashi *et al.* (2005) compared KANBAN, CONWIP and synchronized CONWIP systems for a tree-shaped production process with respect to two performance measures, WIP and backlog, in supply chains consist of assembly stages with different lead times. They considered a supply chain system with three stages, which assembles and supplies one type product. The product is assembled from two distinctive subassemblies, and each subassembly is made up of two distinctive parts. The product is assembled from one unit of each subassembly, and each of the subassemblies is fabricated by using one unit of each part. Their simulation results show the superiority of both CONWIP and synchronized CONWIP over KANBAN, when all inventory levels among the three stages are equally important. The AIDs of this model under CONWIP and KANBAN are depicted in Figure 6 (a) and (b), respectively.

The CONWIP system in Figure 6(a) is specified as follows. There are eight activities in the production process which are deployed as a routing for a final product, including a delivery activity $p_d$, for a warehouse. Let $p_{ij}$ be any one of $p_{11}$, $p_{12}$, $p_{21}$ and $p_{22}$. As like the serial line cases, we assume enough raw material so that we do not have to take care of it. In order to start processing for $p_{ij}$, more than one token should exist in each of $w_{ij}$ (actors) and the respective card buffer. When it starts, one token is decreased from each of them, and one token is in processing in $p_{ij}$. When $p_{ij}$ finishes, the token in $p_{ij}$ is removed, and one token is added to each of $b_{ij}$ and $w_{ij}$. The $p_1$ and $p_2$ processes behave similarly with the corresponding inputs and outputs. The delivery process $p_d$ starts when more than one token exist in each of $b$ and $w_d$. It outputs one token into each of $C_1$, $C_2$, $C_3$, $C_4$ at the commencement. When it finishes, one token is added into $w_d$. The processing times of $p_{ij}$ and $p_k$ are denoted by $h_{ij}$ and $h_k$, respectively.

The KANBAN in Figure 6(b) is now specified. Let $p_{ij}$ be any one of $p_{11}$, $p_{12}$, $p_{21}$ and $p_{22}$. In order to start processing for $p_{ij}$, more than one token should exist in each of $w_{ij}$ and $k_{ij}$. When it starts, one token is decreased from each of them, and one token is being processed in $p_{ij}$. We assume enough raw material as well. After its holding time, when it finishes the processing, the token in $p_{ij}$ is removed, and one token is added to each of $b_{ij}$ and $w_{ij}$. The process $p_1$ starts when more than one token exist in each of the input $b_{11}$, $b_{12}$, $w_1$ and $k_1$. When $p_1$ starts, those tokens are respectively removed, one token is added in each of $k_{11}$ and $k_{12}$, and one token is being processed in $p_1$. When $p_1$ finishes, one token is added in each of $b_1$ and $w_1$. The activities $p_2$ and $p$ work similarly. The delivery process $p_d$ starts when more than one token exist in each of $b$ and $w_d$. It outputs one token into $k$ at the commencement, and one token in $w_d$ at the end of its process.



(a) CONWIP system



(b) Kanban system

Figure 6. Tree-shaped production process with CONWIP and KANBAN

We show four examples of state transition of a tree-shaped production process. Three of them are CONWIP (Figure 6(a)), while the other is KANBAN (Figure 6(b)). Since the number of tokens on a circuit remains the same at any state transition, we can control the WIP on a circuit by initial placement of tokens. Initial tokens on circuits decide the throughput and optimality of the system WIP. All of the following four cases have minimum WIP with respect to throughput.

Case **Tree-CONWIP-1**. Initial inventory for every part is set to zero. Initial deployments of cards are $|C_1| = |C_2| = |C_3| = 2$, and $|C_4| = 3$. Respective numbers of actors are one, except that number of actors for $p_{22}$ that is 2. The whole system shows cyclic behavior with the period 25 [TU] as Table 3 shows. In the table, actor-queues are omitted, because if an activity $p_{11}$, for example, is in processing, then $w_{11}$ is zero. That is, $w_{ij}$ can be think of the opposite of processing, and this makes the table concise.

Case **Tree-CONWIP-2**. Now, we increase one card in $C_2$. That is, initial cards are $|C_1| = |C_3| = 2$, and $|C_2| = |C_4| = 3$. The whole system shows cyclic behavior with the period 12 [TU] as Table 4 shows. Both the former and this case show a complicated situation in finding the optimal deployment of cards for CONWIP on the same production process. When we increase WIP, for example, the former critical circuit becomes non-critical and other circuit is critical with different throughput, and this WIP is still minimum to attain the throughput.

Case **Tree-CONWIP-3**. The respective processing times of $p_{21}$ and $p_{22}$ have been changed here. This case will be used later for comparison between CONWIP and KANBAN. Initial inventory for every part is set to zero. Initial cards are $|C_1| = 2$, $|C_2| = |C_3| = 3$, and $|C_4| = 4$. Respective number of actors remains the same as the former two cases. The whole system shows cyclic behavior with the period 12 [TU] as well, as Table 5 shows.

Table 3. State transition of Tree-CONWIP-1 for a period.

| time | $C_1$ | $p_{11}$ | $b_{11}$ | $C_2$ | $p_{12}$ | $b_{12}$ | $p_1$ | $b_1$ | $C_3$ | $p_{21}$ | $b_{21}$ | $C_4$ | $p_{22}$ | $b_{22}$ | $p_2$ | $b_2$ | $p$ | $b$ | $p_d$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 82 | 0 | 1(8) | 0 | 0 | 1(11) | 0 | 1(7) | 0 | 0 | 1(6) | 0 | 0 | 1(15), 1(2) | 0 | 1(5) | 0 | ---- | 0 | 1(12) |
| 84 | 0 | 1(6) | 0 | 0 | 1(9) | 0 | 1(5) | 0 | 0 | 1(4) | 0 | 0 | 1(13), ---- | 1 | 1(3) | 0 | ---- | 0 | 1(10) |
| 87 | 0 | 1(3) | 0 | 0 | 1(6) | 0 | 1(2) | 0 | 0 | 1(1) | 0 | 0 | 1(10), ---- | 1 | ---- | 1 | ---- | 0 | 1(7) |
| 88 | 0 | 1(2) | 0 | 0 | 1(5) | 0 | 1(1) | 0 | 0 | ---- | 0 | 0 | 1(9), ---- | 0 | 1(12) | 1 | ---- | 0 | 1(6) |
| 89 | 0 | 1(1) | 0 | 0 | 1(4) | 0 | ---- | 0 | 0 | ---- | 0 | 0 | 1(8), ---- | 0 | 1(11) | 0 | 1(5) | 0 | 1(5) |

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 90 | 0 | ---- | 1 | 0 | 1(3) | 0 | ---- | 0 | 0 | ---- | 0 | 0 | 1(7), ---- | 0 | 1(10) | 0 | 1(4) | 0 | 1(4) |
| 93 | 0 | ---- | 0 | 0 | ---- | 0 | 1(9) | 0 | 0 | ---- | 0 | 0 | 1(4) , ---- | 0 | 1(7) | 0 | 1(1) | 0 | 1(1) |
| 94 | 0 | 1(8) | 0 | 0 | 1(11) | 0 | 1(8) | 0 | 0 | 1(6) | 0 | 0 | 1(3), 1(15) | 0 | 1(6) | 0 | ---- | 0 | 1(12) |
| 97 | 0 | 1(5) | 0 | 0 | 1(8) | 0 | 1(5) | 0 | 0 | 1(3) | 0 | 0 | ----, 1(12) | 1 | 1(3) | 0 | ---- | 0 | 1(9) |
| 100 | 0 | 1(2) | 0 | 0 | 1(5) | 0 | 1(2) | 0 | 0 | ---- | 0 | 0 | ----, 1(9) | 0 | 1(12) | 1 | ---- | 0 | 1(6) |
| 102 | 0 | ---- | 1 | 0 | 1(3) | 0 | ---- | 0 | 0 | ---- | 0 | 0 | ----, 1(7) | 0 | 1(10) | 0 | 1(5) | 0 | 1(4) |
| 105 | 0 | ---- | 0 | 0 | ---- | 0 | 1(9) | 0 | 0 | ---- | 0 | 0 | ----, 1(4) | 0 | 1(7) | 0 | 1(2) | 0 | 1(1) |
| 106 | 0 | ---- | 0 | 0 | ---- | 0 | 1(8) | 0 | 0 | ---- | 0 | 0 | ----, 1(3) | 0 | 1(6) | 0 | 1(1) | 0 | ---- |
| 107 | 0 | 1(8) | 0 | 0 | 1(11) | 0 | 1(7) | 0 | 0 | 1(6) | 0 | 0 | 1(15), 1(2) | 0 | 1(5) | 0 | ---- | 0 | 1(12) |

Table 4. State transition of Tree-CONWIP-2 for a period.

| time | $C_1$ | $p_{11}$ | $b_{11}$ | $C_2$ | $p_{12}$ | $b_{12}$ | $p_1$ | $b_1$ | $C_3$ | $p_{21}$ | $b_{21}$ | $C_4$ | $p_{22}$ | $b_{22}$ | $p_2$ | $b_2$ | $p$ | $b$ | $p_d$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 80 | 0 | 1(8) | 0 | 0 | 1(11) | 1 | 1(5) | 0 | 0 | 1(6) | 0 | 0 | 1(15), 1(3) | 0 | 1(7) | 0 | ----- | 0 | 1(12) |
| 83 | 0 | 1(5) | 0 | 0 | 1(8) | 1 | 1(2) | 0 | 0 | 1(3) | 0 | 0 | 1(12), ----- | 1 | 1(4) | 0 | ----- | 0 | 1(9) |
| 85 | 0 | 1(3) | 0 | 0 | 1(6) | 1 | ----- | 1 | 0 | 1(1) | 0 | 0 | 1(10), ----- | 1 | 1(2) | 0 | ----- | 0 | 1(7) |
| 86 | 0 | 1(2) | 0 | 0 | 1(5) | 1 | ----- | 1 | 0 | ----- | 1 | 0 | 1(9), ----- | 1 | 1(1) | 0 | ----- | 0 | 1(6) |
| 87 | 0 | 1(1) | 0 | 0 | 1(4) | 1 | ----- | 0 | 0 | ----- | 0 | 0 | 1(8), ----- | 0 | 1(12) | 0 | 1(5) | 0 | 1(5) |
| 88 | 0 | ----- | 0 | 0 | 1(3) | 0 | 1(9) | 0 | 0 | ----- | 0 | 0 | 1(7), ----- | 0 | 1(11) | 0 | 1(4) | 0 | 1(4) |
| 91 | 0 | ----- | 0 | 0 | ----- | 1 | 1(6) | 0 | 0 | ----- | 0 | 0 | 1(4), ----- | 0 | 1(8) | 0 | 1(1) | 0 | 1(1) |
| 92 | 0 | 1(8) | 0 | 0 | 1(11) | 1 | 1(5) | 0 | 0 | 1(6) | 0 | 0 | 1(3), 1(15) | 0 | 1(7) | 0 | ----- | 0 | 1(12) |

Table 5. State transition of Tree-CONWIP-3 for a period.

| time | $C_1$ | $p_{11}$ | $b_{11}$ | $C_2$ | $p_{12}$ | $b_{12}$ | $p_1$ | $b_1$ | $C_3$ | $p_{21}$ | $b_{21}$ | $C_4$ | $p_{22}$ | $b_{22}$ | $p_2$ | $b_2$ | $p$ | $b$ | $p_d$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 97 | 0 | 1(8) | 0 | 0 | 1(11) | 1 | 1(5) | 0 | 0 | 1(8) | 1 | 0 | 1(8), 1(20) | 1 | 1(7) | 0 | ----- | 0 | 1(12) |
| 102 | 0 | 1(3) | 0 | 0 | 1(6) | 1 | ----- | 1 | 0 | 1(3) | 1 | 0 | 1(3), 1(15) | 1 | 1(2) | 0 | ----- | 0 | 1(7) |
| 104 | 0 | 1(1) | 0 | 0 | 1(4) | 1 | ----- | 0 | 0 | 1(1) | 0 | 0 | 1(1), 1(13) | 0 | 1(12) | 0 | 1(5) | 0 | 1(5) |
| 105 | 0 | ----- | 0 | 0 | 1(3) | 0 | 1(9) | 0 | 0 | ----- | 1 | 0 | -----, 1(12) | 1 | 1(11) | 0 | 1(4) | 0 | 1(4) |
| 108 | 0 | ----- | 0 | 0 | ----- | 1 | 1(6) | 0 | 0 | ----- | 1 | 0 | -----, 1(9) | 1 | 1(8) | 0 | 1(1) | 0 | 1(1) |
| 109 | 0 | 1(8) | 0 | 0 | 1(11) | 1 | 1(5) | 0 | 0 | 1(8) | 1 | 0 | 1(20), 1(8) | 1 | 1(7) | 0 | ----- | 0 | 1(12) |

Now, we apply the theory of this paper to the above cases. First consider Tree-CONWIP-1. By Proposition 6, the throughput of a critical circuit is that of the whole system. Denote the circuit $C_2p_{12}b_{12}p_1b_1pbC_2$ as $\overline{C}_2$. This circuit is critical. In fact, by observing the state transition table, we see any token on $\overline{C}_2$ is not blocked. Then, by

Proposition 5, $\overline{C}_2$ is critical. Every activity on $\overline{C}_2$ starts twice in a period. Therefore, the throughput of $\overline{C}_2$ and the whole system is 2/25 [PC/TU].

In Tree-CONWIP-2 and Tree-CONWIP-3 cases, the critical circuit is $p_2 w_2 p_2$; that is, $p_2$ is a critical activity. The period is 12 [TU], and the throughput is 1/12 [PC/TU]. The respective placements of system WIP are optimum for this throughput. The system WIP of Tree-CONWIP-2 is 10.33, while that of Tree-CONWIP-3 is 11.75. For the former case, for example, we show how to calculate the system WIP by using the state transition table, according to the definition of average WIP given in Section 3.1. Consider Table 4. We can count tokens in the places of activity circuits and those in the rest of the places, separately. The number of tokens on an activity circuit is unchanged from the initial state, which is the number of actors for the activity. On the activity circuit $p_{22}$, for example, there are two tokens. It means that the WIP on this activity circuit is 2. Here, let us denote the sum of WIP on all of activity circuits by $W_A$. Since we have 8 activities, and since $p_{22}$ has two actors, we have $W_A = 9$. Now, let us count tokens in the rest of the places, which are actually connecting queues. Take $b_{12}$ as such an example. By observing the state transition table for a period from time 88 through 91, a token remains in $b_{12}$ is for 9 [TU]. Thus, its integration value for a period is $1 * 9 = 9$ [PC*TU]. Denote the average WIP in all of the connecting queues by $W_Q$. Since the connecting queues are $C_1, b_{11}, C_2, b_{12}, b_1, C_3, b_{21}, C_4, b_{22}, b_2, b$, we can calculate respective integration values for a period $L$ from the table, and add them. Thus, we have $W_Q \cdot L = (0+0+0+9+2+0+1+0+4+0+0) = 16$. Therefore, the system WIP is $W_A + W_Q = 9 + (16/12) = 10.33$.

An example of KANBAN for Tree-shaped process is as follow.

Case **Tree-KANBAN**. Initial inventory for every part is set to zero. Initial cards are set as $k_{11} = k_{12} = k_{21} = k_1 = k_2 = k = 1$ and $k_{22} = 2$. Respective numbers of actors are one, except that number of actors for $p_{22}$ that is 2. The state transition is given in Table 6, where the throughput is 1/12.

The connecting queues are $k_{11}, b_{11}, k_{12}, b_{12}, k_1, b_1, k_{21}, b_{21}, k_{22}, b_{22}, k_2, b_2, k$ and $b$. Thus, by calculating respective WIP for those queues from Table 6, we have

$W_Q \cdot L = (0+4+0+1+0+3+0+4+0+4+0+0+7+0) = 23$

Therefore, the system WIP is

$W_A + W_Q = 9 + (23/12) = 10.92$

Table 6. State transition of Tree-KANBAN for a period.

| time | $k_{11}$ | $p_{11}$ | $b_{11}$ | $k_{12}$ | $p_{12}$ | $b_{12}$ | $k_1$ | $p_1$ | $b_1$ | $k_{21}$ | $p_{21}$ | $b_{21}$ | $k_{22}$ | $p_{22}$ | $b_{22}$ | $k_2$ | $p_2$ | $b_2$ | $k$ | $p$ | $b$ | $p_d$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 109 | 0 | 1(3) | 0 | 0 | 1(6) | 0 | 0 | 1(4) | 0 | 0 | 1(3) | 0 | 0 | 1(3), 1(15) | 0 | 0 | 1(7) | 0 | 1 | ---- | 0 | 1(12) |
| 112 | 0 | ---- | 1 | 0 | 1(3) | 0 | 0 | 1(1) | 0 | 0 | ---- | 1 | 0 | ----, 1(12) | 1 | 0 | 1(4) | 0 | 1 | ---- | 0 | 1(9) |
| 113 | 0 | ---- | 1 | 0 | 1(2) | 0 | 0 | ---- | 1 | 0 | ---- | 1 | 0 | ----, 1(11) | 1 | 0 | 1(3) | 0 | 1 | ---- | 0 | 1(8) |
| 115 | 0 | ---- | 1 | 0 | ---- | 1 | 0 | ---- | 1 | 0 | ---- | 1 | 0 | ----, 1(9) | 1 | 0 | 1(1) | 0 | 1 | ---- | 0 | 1(6) |
| 116 | 0 | 1(8) | 0 | 0 | 1(11) | 0 | 0 | 1(9) | 0 | 0 | 1(8) | 0 | 0 | 1(20), 1(8) | 0 | 0 | 1(12) | 0 | 0 | 1(5) | 0 | 1(5) |
| 121 | 0 | 1(3) | 0 | 0 | 1(6) | 0 | 0 | 1(4) | 0 | 0 | 1(3) | 0 | 0 | 1(15), 1(3) | 0 | 0 | 1(7) | 0 | 1 | ---- | 0 | 1(12) |

The following proposition partly resolves the dynamics of the tree-shaped production control.

**Proposition 9.** Consider the tree-shaped production process shown in Figure 6 with CONWIP and KANBAN. Assume that both systems have the same actors for respective processes, the same activation frequency, and the same throughput. Let $N$ and $K$ be the total number of cards in CONWIP and KANBAN, respectively. Then, we have the following.

(i) If $N - K \leq \dfrac{h_1 + h_2 + 3h}{\lambda}$, then $W_C \leq W_K$,

(ii) If $N = K$, then $W_C < W_K$,

where $\lambda$ is the maximum cycle mean, and $W_C$ and $W_K$ are the average system WIP of CONWIP and KANBAN, respectively.

For the tree-shaped production process in the above proposition, many CONWIP and KANBAN cases, which have the same level of throughput, satisfy the if-condition of (i), and then $W_C < W_K$ certainly holds. However, the if-condition is not satisfied by Tree-CONWIP-3 and Tree-KANBAN, where the system WIP is 11.75 and 10.92, respectively. That is, the if-condition of the Proposition 9 is meaningful, and we would say that CONWIP does not necessarily outperform KANBAN.

The statement on comparison between CONWIP and KANBAN is different to Takahashi *et al.* (2005). Our result requires finite capacity, while Takahashi *et al.* (2005) considered infinite capacity case. As we have shown in Section 3, the optimality of system WIP requires analysis of critical circuit in the system. The same number of system WIP, in general, does not assure us the optimality, because the placement of tokens changes the best possible throughput and the corresponding minimum amount of system WIP. By specifying production process and placement of WIP, we can conduct the analysis of comparison of best tuned performance of CONWIP and KANBAN as like in Proposition 9.

## 5. Conclusion

By employing the framework of token transaction system, this paper showed how and for what the Little's law holds with respect to average WIP, average cycle time, and average throughput. In order to apply the framework, the target production control systems should have deterministic processing time, a strongly connected structure, and connecting queues with FIFO control policy. Since such token transaction systems show periodic behavior, we can design dynamic properties of production processes, which are related to the Little's law.

The method developed in this paper has revealed the following issues.

(1) WIP

The Little's law holds on every circuit in a production process modeled as a token transaction system. The sum of average WIP in the whole system does not necessarily give us any relation to the cycle time or throughput of the system. Or, focusing on inventory in a single storage or warehouse does not show the Little's law, neither. Since the law is much fundamental as a physical law of material logistics in a factory and wider logistics network, this may bring a strong impact on the way to measure the amount of inventories in production control research.

In general, there is a trade-off between amount of WIP and lead time (i.e., cycle time). That trade-off should be considered by focusing on proper circuit, according to the theory. Otherwise, the analysis might be very vague, because there could not be a proper relation between those indices. Furthermore, WIP should not restricted to be tangible inventory. From the token transaction systems theoretic point of view, production orders and production resources work as WIP in the sense that they decide whether an activity in the process can start or not. The whole configuration of those three kinds of WIP bring out the resultant performance of the whole process.

(2) Throughput

The throughput of the whole process is decided by a critical circuit in the process. A critical circuit is formed as the result of the connecting structure of activities and queues and the deployment of WIP. As stated in (1) above, WIP in a token transaction system represent inventory of part or material, production capacity, and production orders.

(3) Comparison of CONWIP and KANBAN

CONWIP is superior to KANBAN in some cases, while it is not in other cases. Superiority here refers the fact that the minimum system WIP is smaller than the other to attain the same throughput by deploying suitable number of cards. As shown in Section 4, there is no universal superiority between CONWIP and KANBAN.

Comparison of production control systems can be complicated. One reason that this paper showed is in the complex relation among WIP deployment, critical circuits, and throughput. Such examples are Tree-CONWIP-1 and Tree-CONWIP-2 cases. Even if we change the number of tokens, the resultant WIP can be still optimum in the sense that the WIP is minimum to attain the changed throughput.

There are some related topics remained. Effect of randomness needs to be considered. Basic question is: How will randomness affect the criticality of critical circuits? Original idea of CONWIP does not restrict to FIFO policy. Sophisticated policy may lead the process to different performance.

## References

Baccelli, F.L., Cohen, G., Olsder, G.J. and Quadrat, J.P, 1992. *Synchronization and Linearity -- An algebra for discrete event systems.* John Wiley.

Bonvik, A.M. and Gershwin, S.B., 1996. Beyond Kanban: Creating and analyzing lean shop floor control policies. *Manufacturing and Service Operations Management Conference Proceeding,* Durtmouth College, The Amos Tuck School, Hannover, New Hampshire, 46-51.

Bonvik, A.M., Couch, C.E. and Gershwin, S.B., 1997. A comparison of production-line control mechanisms. *International Journal of Production Research,* 35(3), 789–804.

Framinan, J.M., Gonzalez, P.L. and Ruiz-Usano, R., 2003. The CONWIP production control system: Review and research issues. *Production Planning and Control*, 14, 255–265.

Gstettner, S. and Kuhn, H., 1996. Analysis of production control systems kanban and CONWIP. *International Journal of Production Research,* 34 (11), 3253–3274.

Hopp, W.J. and Spearman, M.L., 2001. *Factory physics: foundations of manufacturing management.* Irwin/McGraw-Hill.

Little. J.D.C., 1961. A proof for the queuing formula: $L = \lambda W$ . *Operations Research*, 9, 383-387.

Mesarovic, M. D. and Takahara, Y., 1975. *Mathematical foundation of general systems theory*. Academic.

Monden, Y., 1998. *Toyota production system: an integrated approach to just-in-time.* 3rd ed. Engineering & Management Press.

Muckstadt, J.A. and Tayur, S.R., 1995. A comparison of alternative kanban control mechanisms: I, background and structural results. *IIE Transactions*, 27 (1), 140–150.

Murata, T., 1992. *Analysis and Application of Petri nets.* Kindai Kagakusha. (in Japanese)

Paternina-Arboleda C.D. and Das, T.K., 2001. Intelligent dynamic control policies for serial production lines. *IIE Transactions*, 33(1), 65–77.

Sato, R., 2001. Realization theory of discrete-event systems and its application to the uniqueness and universality of DEVS formalism. *International Journal of General Systems*, 30(5), 513-549.

Sato, R. and Kawai, A., 2007. Organizing a business process that realizes required throughput: the principle and an application to information systems for SCM. Department of Social Systems and Management, Discussion Paper Series No.1184, University of Tsukuba.

Sato, R. and Praehofer H., 1997. A discrete event model of business system - A Systems Theoretic for Information Systems Analysis: Part 1. *IEEE Trans. Systems, Man, and Cybernetics,* Volume 27, 1-10.

Spearman, M.L. and Zazanis, M.A., 1992. Push and pull production systems: issues and comparisons. *Operations Research*, 40, 521–532.

Spearman, M.L., Woodruff, D.L. and Hopp, W.J., 1990. CONWIP: A pull alternative to Kanban. *International Journal of Production Research*, 23, 879–894.

Takahashi, K., Myreshka, and Hirotani, D., 2005. Comparing CONWIP, synchronized CONWIP, and Kanban in complex supply chains. *International Journal of Production Economics*, 93-94, 25-40.

Yang, K.K., 2000. Managing a flow line with single-Kanban, dual-Kanban or CONWIP. *Production and Operations Management*, 9(4), 349-366.

Zeigler, B. P., 1976. *Theory of modelling and simulation.* John Wiley.