Hindawi
Mathematical Problems in Engineering
Volume 2018, Article ID 9189836, 9 pages
https://doi.org/10.1155/2018/9189836

Research Article

Modeling and Verification of Reconfigurable Printing System Based on Process Algebra

Rubai Luo[1,2] **Shasha Gao**[1,2] **Huailin Li**[2,3] **and Shisheng Zhou**[1,2,3]

[1]*Faculty of Printing, Packing Engineering, and Digital Media Technology, Xi'an University of Technology, Xi'an, Shaanxi 710048, China*
[2]*Shaanxi Provincial Key Laboratory of Printing and Packaging Engineering, Xi'an University of Technology, Xi'an 710048, China*
[3]*School of Mechanical and Precision Instrument Engineering, Xi'an University of Technology, Xi'an 710048, China*

Correspondence should be addressed to Huailin Li; lihuailin6666@sina.com

Received 26 March 2018; Revised 6 August 2018; Accepted 18 September 2018; Published 4 October 2018

Academic Editor: Jean Jacques Loiseau

With the print production process as the research object in this paper, the intelligent-context-reconfigurable printing system model is analyzed using process algebra (PA). First, combined with the printing production process, the overall framework of the system model, based on the agent-resource-worker (ARW) component model, is proposed. Abstract and formal verification of the system model are then carried out, and the verification process of the complex calibration process is discussed. Finally, the security and progress attributes of the model are validated by mCRL2. The results show that the system model has good reliability.

1. Introduction

As customer demand becomes increasingly varied, demand for product customization and pressure for the rapid introduction of new products are increasing [1]. Traditional manufacturing systems, such as dedicated and flexible systems, cannot provide an adequate response at a reasonable cost and have significant shortcomings in meeting these requirements [2]. Flexible manufacturing systems can convert between different production variants, but they introduce issues of limited flexibility, low productivity, and low return on investment (ROI) [3, 4]. Yoram Koren, a professor who proposed the concept of reconfigurable manufacturing systems for the first time in 1999, thought that reconfigurable manufacturing systems could accurately provide the required functionality [5]. In particular, the modular structure of reconfigurable manufacturing systems enables them to shorten the time to design, build, and redesign a system [6]. Therefore, it is widely thought that reconfigurable manufacturing systems (RMSs) will be very common in the future [7].

Reconfigurability is an engineering technology that can help realize rapid, cost-effective responses to market and product changes [8]. Systematic and continuous design and acceleration are the cornerstones of the RMS concept and will appear often throughout its life cycle [6]. Much research has been done on problems related to RMSs [9–16]. The reconfigurability of a printing system is mainly reflected in its ability to change production paths based on product requirements and related environmental changes. Here, "environment" mainly refers to the status of the printing press, material reserves, changes in material interaction and temperature, and humidity in the workshop. This article discusses the design of a contextualized intelligent printing system that can intelligently, dynamically, and quickly configure and organize all kinds of manufacturing systems based on different printed product features.

Process algebra (PA) is a formal modeling tool for describing complex systems, and its purpose is to provide a method for studying concurrency and interaction [17]. Process algebra describes the model in algebraic form and defines a complete set of syntax and semantics for the modeled system [18]. The algebraic method based on calculus is now used more in the field of software reusability with fewer applications in RMS [19]. Wei Yaya et al. first attempted to use process algebra to describe a workflow, which provided feasibility verification for manufacturing systems [20]. In addition to using the semantics defined in process algebra and computing rules to verify the viability of a system, the software can be used to validate the reliability of a system [21].

The second section of this article introduces operators related to process algebra. The third section discusses the printing system and explains its modeling ideas and workflow. Section 4 makes use of process algebra for more complex parts of a system for reasoning and verification, and it is further verified by mCRL2. Section 5 summarizes this article.

2. The Introduction of Process Algebra

Process algebra is a high-level algebraic description language that supports the combined description of concurrency, distributed systems and their formal proof of form [17]. It describes the models algebraically, combining subsystems with operators to hide system-unconscious activities of hidden operators, as internal activities and defining a complete set of syntax and semantics for the modeled system. A complete description of process algebra can be found in several references [17, 18, 21–26]. Process Algebra is defined as follows:

① Assume a finite, nonempty set A of (atomic) actions, representing indivisible behavior (such as reading or sending a datum). Each atomic action is a constant that can execute itself, after which it terminates successfully:

$$a \xrightarrow{a} \surd \tag{1}$$

② "+" is called alternative composition. The closed term $t_1 + t_2$ represents the process that executes either t_1 or t_2.

③ "·" is called sequential composition. The closed term $t_1 \bullet t_2$ represents the process that executes first t_1 and then t_2.

④ "||" is called a merge. This binary operator executes the two process terms in its arguments in parallel. That is, $s||t$ can choose to execute an initial transition of s, an initial transition of t, or communication between s and t.

⑤ "|" is called a communication merge. The operation s|t executes as initial transition a communication between initial transitions of the process terms s and t and then behaves as the standard merge operator ||.

⑥ "$\between$" is called a left merge. The left merge $s \between t$ takes its initial transition from the process term s and then behaves as the merge ||.

⑦ "δ" is called deadlock, and it displays no behavior. The communication function γ is extended by allowing the communication of two atomic actions and results in δ, that is, $\gamma : A \times A \longrightarrow A \cup \{\delta\}$. This extension of γ enables us to express that two actions do not communicate by defining $\gamma(a, b) \triangleq \delta$.

⑧ "∂" is the unary encapsulation operator. The $\partial_H(t)$ can execute all transitions of t whose labels are not in H.

⑨ "τ" is a unary abstraction operator, which enables us to abstract away from the internal computation steps of an implementation. For example, $\tau_I(t)$ expresses that in $\tau_I(t)$, all of the labels of transitions of t that are in I are renamed into τ.

3. Reconfigurable Printing System Model

Reconfigurable manufacturing systems were first designed to quickly change hardware and software component structures to allow quick adjustment of the production capacity and function of a family of parts due to sudden changes in production requirements [9].

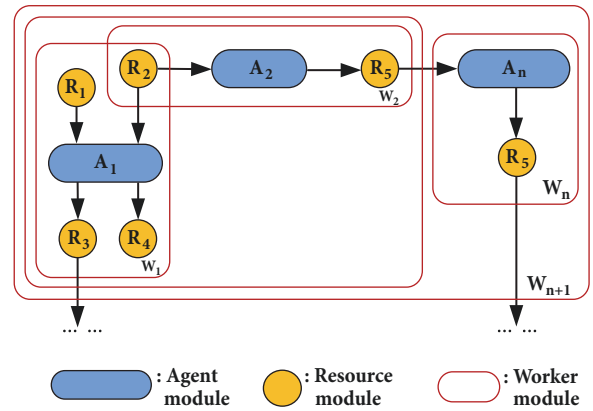

Figure 1: ARW component model.

The basic process of a printing system is as follows.

(1) Prepress stage: process the graphic original to obtain production data that controls the printing and postpress stages.

(2) Printing stage: select the appropriate printing process according to production data generated by prepress to transfer the ink, varnish, and other materials to the substrate material surface (such as paper, fabric, or leather) after physical processing and getting the sheet.

(3) Postpress stage: apply finishing processes (such as light, hot stamping, die-cutting, and indentation) to the sheet to produce the desired product. After assembly-adaptive processing (such as cutting and folding) the final product is assembled after parts processing (such as stickers, plastic backing, and binding).

In the modeling process, the agent-resource-worker (ARW) component is taken as the basic model, and the functions, attributes, and behavior features of the component are abstracted to build agent, resource, and worker models. An agent model describes the type of agent, structure, and realization of the function. A resource model is a collection of various job definition format (JDF) resources. The worker model is an integrated model of the agent and resource models, embodying the final behavior of the ARW component model, as shown in Figure 1.

Worker models will form a new worker model in different composite ways, with the main composition shown in Figure 2, where a is sequential compounding, b is feedback compounding, c is collaborative compounding, d is compounding selection, and e through g are three-component compounds, which can be abstracted as one component and one bicomponent mechanism compound. This can be extended to multiple-component compound mechanisms. In modeling, agents are categorized as executive agents and decision agents. The difference between them is mainly that a decision agent has a context-aware function, which can make decisions according to changes in the environment. An executive-class agent can independently perform its own set

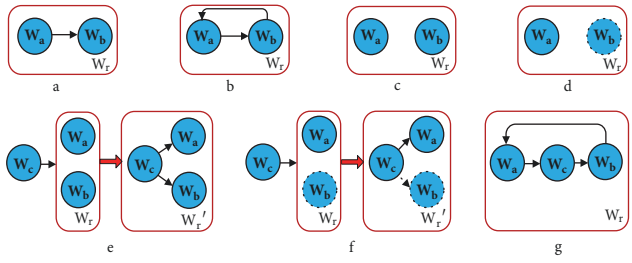

Figure 2: Composite approach of worker model.

of functions. That is, when the agent receives data, it can process it accordingly and then send the processed data to the next agent.

As shown in Figure 3, each execution-class agent is named by its function. For example, page making means that the agent is making a page. The workflow simulated by this model is as follows. In the prepress, the first processing of the original is carried out according to the needs of the customer, and the processed file is proofed. It is one step in the printing process. The next step is proofreading. The proofreading is mainly to verify whether the sample is qualified. If the sample is qualified, then the plate is made. Otherwise, the proofing is done again, and the maximum number of samples is n_{proof}. In the printing stage, according to the relevant data obtained in the prepress stage, the decision agent chooses the appropriate printing press and judges whether the materials and workshop environment are suitable for printing and drying. (In the model, *No*.1 is used to represent the key parts of the intelligent printing system in the context of the different printing presses, which is not described extensively in this article.) In the postpress phase, the decision agent selects the appropriate postprinting procedure according to the preprinted data, and the corresponding agent is executed until the finished product is obtained.

4. Reasoning and Verification

In the reconfigurable printing system model of Figure 3, the real line box is mainly aimed at the characteristics of the bounded retransmission communication protocol for the large-data-volume transmission to meet the needs of repeated proofing in the printing process, control the number of samples, and reduce the cost. Due to space limitations, this paper focuses on the verification of the implementation. The rest of the process is executed sequentially, which can be verified according to the algebraic operations in the second chapter.

4.1. Model Abstraction. The model is abstracted as shown in Figure 4, where A_1, K, and A_2 are the execution agents, C_1 and C_2 are counters, and the number of proofs is controlled. The algebraic expression uses the following data parameters and functions: d ranges over a finite data set $\triangle$, and b ranges over $\{0, 1\}$, while n ranges over $\{0, \ldots, max\}$, where max is used to limit the number of transmissions. If $max = 2$, the maximum number of transmissions is three. Finally, *ack* denotes an acknowledgement message, *to* denotes count-out, and $\perp$ is a data-error message.

In this model, A_1, K, and A_2 are three agents in the system. The algebraic expression of A_1 is

$$\begin{aligned} &X(b,n) = \sum_{d\epsilon\triangle} r_1(d) \bullet Y(d,b,n) \\ &Y(d,b,n) = s_2(d,b,n) \bullet Z(d,b,n) \\ &(n < \max) \\ &Z(d,b,n) = r_5(ack) \bullet X(1-b,n) + r_5(\perp) \\ &\quad \bullet Y(d,b,n+1) \\ &Z(d,b,\max) = r_5(ack) \cdot X(1-b,n) + r_5(\perp) \bullet r_6(to) \\ &\quad \cdot s_7(to) \cdot STOP \end{aligned} \tag{2}$$

The algebraic expression of K is

$$\begin{aligned} &V(b,n) = \sum_{d\epsilon\triangle} r_2(d,b,n) \cdot W(d,b,n) \\ &(n < max) \\ &W(d,b,n) = s_3(d,b,n) \cdot V(1-b,n) + s_3(\perp) \cdot V(b,n) \\ &W(d,b,max) = s_3(d,b,n) \cdot V(1-b,n) + s_3(\perp) \\ &\quad \cdot s_6(to) \cdot V(b,n) \end{aligned} \tag{3}$$

The algebraic expression of A_2 is

$$\begin{aligned} &R(b,n) = \sum_{d\epsilon\triangle} r_3(d,b,n) \cdot T(d,b,n) + r_3(\perp) \cdot U(b) \\ &T(d,b,n) = s_4(d,b) \cdot s_5(ack) \cdot R(1-b,n) \\ &U(b) = s_5(\perp) \cdot \left(R(b,n) + r_7(to) \cdot R(b,n)\right) \end{aligned} \tag{4}$$

In state $X(b,n)$, A_1 reads data d from channel 1, after which it proceeds to the $Y(d,b,n)$ state:

$$X(b,n) = \sum_{d\epsilon\triangle} r_1(d) \bullet Y(d,b,n) \tag{5}$$

In state $Y(d,b,n)$, data d is processed by A_1, with bits b and n attached to the data. Then the message (d,b,n) is sent by channel 2:

$$Y(d,b,n) = s_2(d,b,n) \cdot Z(d,b,n) \tag{6}$$

In state $Z(d,b,n)$, A_1 waits for either an acknowledgement or a data-error message ($\perp$) via channel 5. Suppose A_1 receives an acknowledgement from A_2. Then, A_1 will proceed to state $X(1-b,n)$. If A_1 receives a data-error message from channel 5 and the number n is less than the maximum, it proceeds to state $Y(d,b,n+1)$. If $n = max$, then A_1 receives a

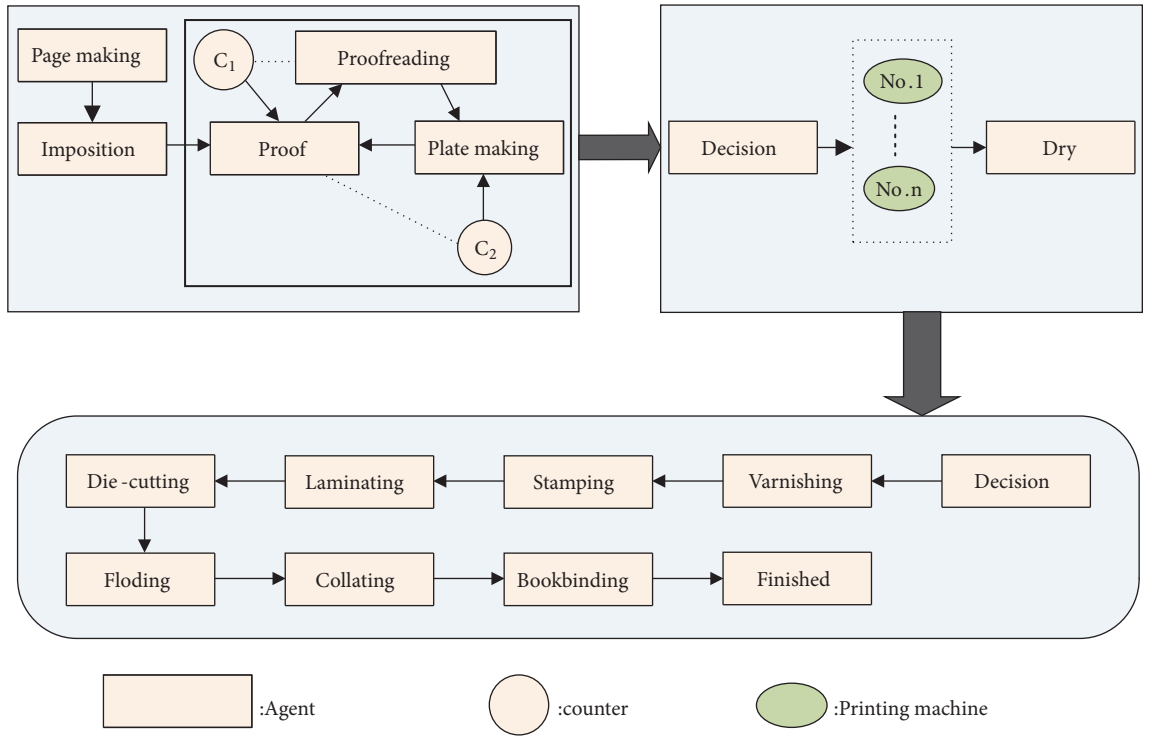

Figure 3: Model of reconfigurable printing system.

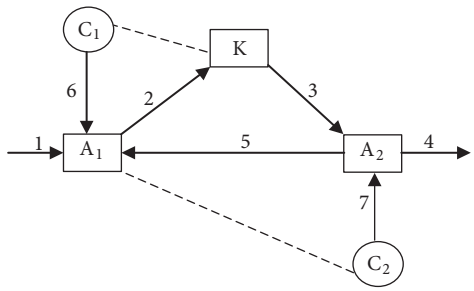

Figure 4: Abstraction of the model.

count-out (to) message from channel 6, and sends a count-out by channel 7. Then, the system issues a warning and implements a manual intervention, which is the *STOP* in the algebraic expression:

$$\begin{aligned}
&(n < max)\\
&Z(d,b,n) = r_5(ack) \bullet X(1-b,n) + r_5(\perp)\\
&\quad \bullet Y(d,b,n+1)\\
&Z(d,b,max) = r_5(ack) \cdot X(1-b,n) + r_5(\perp) \bullet r_6(to)\\
&\quad \cdot s_7(to) \cdot STOP
\end{aligned} \tag{7}$$

In state $V(b,n)$, K receives data from channel 2, then it judges the received data and determines whether it is qualified. Then K turns to state $W(d,b,n)$:

$$V(b,n) = \sum_{d\epsilon\triangle} r_2(d,b,n) \cdot W(d,b,n) \tag{8}$$

In state $W(d,b,n)$, if the data are qualified, K sends a data message via channel 3; otherwise, it sends data-error information through channel 3 if $n < max$. If $n = max$, it sends (to) by channel 6 and turns to state $V(b,n)$:

$$\begin{aligned}
&(n < max)\\
&W(d,b,n) = s_3(d,b,n) \cdot V(1-b,n) + s_3(\perp) \cdot V(b,n)\\
&W(d,b,max) = s_3(d,b,n) \cdot V(1-b,n) + s_3(\perp)\\
&\quad \cdot s_6(to) \cdot V(b,n)
\end{aligned} \tag{9}$$

In the state $R(b,n)$, A_2 waits for the (d,b,n). If it receives such a message, then it proceeds to state $T(d,b,n)$, in which case it will send the data via channel 4, send an acknowledgement message via channel 5, and turn to state $R(b,n)$. If it receives a data-error message ($\perp$) from channel 3, it will proceed to state $U(b)$ to send this message by channel 5 and directly enter the $R(b,n)$ state. It may also receive (to)

information from channel 7. And then it will enter the $R(b,n)$ state and wait for retransmission:

$$\begin{aligned} R(b,n) &= \sum_{d\in\triangle} r_3(d,b,n)\cdot T(d,b,n) + r_3(\perp)\cdot U(b) \\ T(d,b,n) &= s_4(d,b)\cdot s_5(ack)\cdot R(1-b,n) \\ U(b) &= s_5(\perp)\cdot\left(R(b,n) + r_7(to)\cdot R(b,n)\right) \end{aligned} \tag{10}$$

4.2. Verification Process. Suppose the communication between all agents is successfully implemented in this model. The following equations can be derived from ε_{ACP}, commutativity of the merge, and *RDP*. The proof is as follows.

Step 1. Define the communication:

$$\begin{aligned} \gamma(s_2(d,b), r_2(d,b)) &\triangleq c_2(d,b) \\ \gamma(s_3(d,b), r_3(d,b)) &\triangleq c_3(d,b) \\ \gamma(s_3(\perp), r_3(\perp)) &\triangleq c_3(\perp) \\ \gamma(s_5(\perp), r_5(\perp)) &\triangleq c_5(\perp) \\ \gamma(s_5(ack), r_5(ack)) &\triangleq c_5(ack) \\ \gamma(s_6(to), r_6(to)) &\triangleq c_6(to) \\ \gamma(s_7(to), r_7(to)) &\triangleq c_7(to). \end{aligned} \tag{11}$$

Step 2. Define encapsulation and abstraction operators:

$$\begin{aligned} H = {} & \{s_2(d,b), r_2(d,b), s_3(d,b), r_3(d,b) \mid d\in\triangle, b \\ & \in\{0,1\}\}\cup\{s_3(\perp), r_3(\perp), s_5(ack), r_5(ack), s_5(\perp), \\ & r_5(\perp), s_6(to), r_6(to), s_7(to,), r_7(to)\}; \\ I = {} & \{c_2(d,b), c_3(d,b) \mid d\epsilon\triangle, b\in\{0,1\}\}\cup\{c_3(\perp), \\ & c_5(ack), c_5(\perp), c_6(to), c_7(to)\}. \end{aligned} \tag{12}$$

The expression for the entire system is

$$\tau_I\left(\partial_H\left(X(b,n)\parallel V(b,n)\parallel R(b,n)\right)\right). \tag{13}$$

A_1 receives the information d and reads the data to carry on proofing, in which $b=0, n=0$. Proofing is one of the steps in the printing process. And the state of the whole system is

$$\begin{aligned} &\partial_H(X(0,0)\parallel V(0,0)\parallel R(0,0)) \\ &\quad = \sum_{d\in\triangle} r_1(d)\cdot\partial_H(Y(d,0,0)\parallel V(0,0)\parallel R(0,0)) \end{aligned} \tag{14}$$

A_1 then sends data $(d,0,0)$ by channel 2 to K, and K receives it:

$$\begin{aligned} \partial_H(Y(d,0,0)\parallel V(0,0)\parallel R(0,0)) = {} & \sum_{d\in\triangle}\partial_H \\ & \cdot\left(s_2(d,0,0)\cdot Z(d,0,0)\parallel r_2(d,0,0)\right. \\ & \left.\cdot W(d,0,0)\parallel R(0,0)\right) = c_2(d,0,0) \\ & \cdot\partial_H(Z(d,0,0)\parallel W(d,0,0)\parallel R(0,0)) \end{aligned} \tag{15}$$

K is for proofreading. Proofreading means checking whether the samples are qualified.

① If sample is qualified, then K will send $(d,0,0)$ to A_2 and A_2 will receive it.

$$\begin{aligned} &\partial_H(Z(d,0,0)\parallel W(d,0,0)\parallel R(0,0)) \\ &\quad = \sum_{d\in\triangle}\partial_H\left(Z(d,0,0)\left\|s_3(d,0)\cdot V(1,0)\right\| r_3(d,0,0)\right. \\ &\quad \left.\cdot T(d,0,0)\right) = c_3(d,0,0)\cdot\partial_H(Z(d,0,0)\parallel V(1,0)\parallel \\ &\quad T(d,0,0)) \end{aligned} \tag{16}$$

After A_2 receives, it will be sent by channel 4, while A_1 will send a confirmation message.

$$\begin{aligned} &\partial_H(Z(d,0,0)\parallel V(1,0)\parallel T(d,0,0)) = \partial_H\left(r_5(ack)\right. \\ &\quad \left.\cdot X(1,0)\parallel V(1,0)\parallel s_5(ack)\cdot R(1,0)\right) = c_5(ack) \\ &\quad \cdot s_4(d,0)\cdot\partial_H(X(1,0)\parallel V(1,0)\parallel R(1,0)) \end{aligned} \tag{17}$$

② If the sample is not qualified and the number of times is less than *max*, then K sends $(\perp)$ to A_2, and A_2 sends feedback information to A_1. A_1 receives it and reprocesses the original information and $(n+1)$.

$$\begin{aligned} &\partial_H(Z(d,0,0)\parallel W(d,0,0)\parallel R(0,0)) \\ &\quad = c_3(\perp)\cdot c_5(\perp) \\ &\qquad \cdot\partial_H(Y(d,0,1)\parallel V(0,1)\parallel R(0,1)) \\ &\qquad\qquad (n<max) \end{aligned} \tag{18}$$

A sends the reprocessed data to K. The data is checked by K.

$$\begin{aligned} &\partial_H(Y(d,0,1)\parallel V(0,1)\parallel R(0,1)) \\ &\quad = c_2(d,0,1)\cdot\partial_H(Z(d,0,1)\parallel V(0,1)\parallel R(0,1)) \end{aligned} \tag{19}$$

If the max sample is passed, then

$$\begin{aligned} &\partial_H(Z(d,0,max)\parallel W(d,0,max)\parallel R(0,max)) \\ &\quad = c_5(ack)\cdot s_4(d,0) \\ &\qquad \cdot\partial_H(X(1,0)\parallel V(1,0)\parallel R(1,0)) \end{aligned} \tag{20}$$

If the proof fails, the system will stop running

$$\begin{aligned} &\partial_H(Z(d,0,max)\parallel W(d,0,max)\parallel R(0,max)) \\ &\quad = c_3(\perp)\cdot c_5(\perp)\cdot c_6(to)\cdot c_7(to)\cdot STOP \end{aligned} \tag{21}$$

The state transition of $\partial_H(X(1,0)\parallel V(1,0)\parallel R(1,0))$ is the same as that of $\partial_H(X(0,0)\parallel V(0,0)\parallel R(0,0))$. According to the above verification process, the process chart of the model can be drawn as follows, where 1-5 is the state transition when $d=d_1, b=0$, and 5-1 is the state transition when $d=d_2, b=0$.

After application of the abstraction operator τ_I, communication actions over the internal channels 2, 3, 5, 6, and 7 are renamed into τ, after which most of these actions can be removed using axiom $B1$.

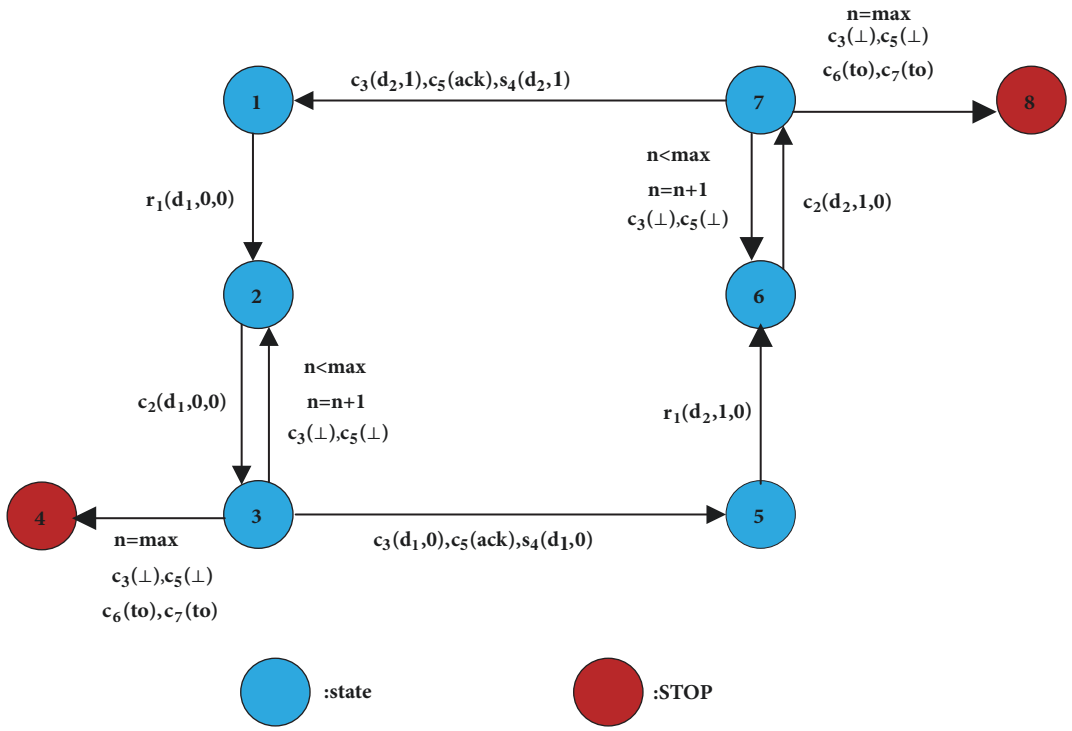

Figure 5: Process diagram of the model.

4.3. Verification with mCRL2. The tools used for process algebra testing mainly include FDR2, the Caesar-Aldebaran toolset (CADP), mCRL2, and LTSA. The mCRL2 has strong interactivity and can intuitively display the migration of each state in the system. Therefore, mCRL2 is used to validate the model in Section 4.1. The mCRL2 is a formal specification language with an associated toolset [26] that supports linearization, simulation, state-space exploration, and generation and has tools to optimize and analyze specifications. Moreover, state spaces can be manipulated, visualized, and analyzed. With this language, users can specify the behavior of a distributed system and analyze it using automated techniques [27].

The process of verification with mCRL2 is as follows.

Step 3. Write the system process using the mCRL2 language.

Step 4. Convert mCRL2 to linear process specification (LPS). Various operating tools can be used to modify or simplify LPS.

Step 5. Convert LPS to a label transition system (LTS) or state space. This error can then be analyzed using the LTS.

Following the above steps, the mcrl2graph tool is used to analyze the generated lts file. The resulting label-migration diagram is shown in Figure 6. And the mCRL2 specification of the process is shown in Appendix B. This tool is used to step through the state-transition process and generate a state-transition diagram, as shown below. Throughout the simulation, $e0$ and $e1$ represent the additional bits, an $max = 1$, which means that the maximum number of transmissions

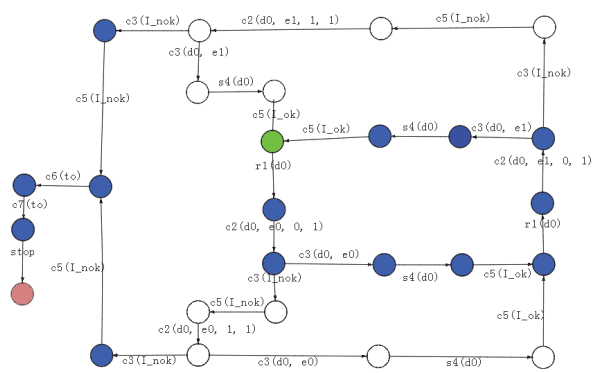

Figure 6: State-transition diagram of the system.

is two. The state-transition diagram obtained by simulation using mCRL2 is consistent with the system state-migration process of inference in Section 4.2. Among them, the green state point indicates the initial state of the system. At the same time, the repeated transmission part (between state 2 and 3 and state 6 and 7) in Figure 5 is expanded in detail and represented by state point without any color.

5. Conclusions

Based on process algebra, this paper constructs a reconfigurable printing system model using an ARW component model. By abstracting and verifying the system model using process algebra, the inference results show that the system can

fulfill the expected functions. Using mCRL2's security and schedule attributes on the system model, its execution is in line with the actual printing process and has good reliability.

Appendix

A.

(1) Axioms for BPA

$A1$: $x + y = y + x$;
$A2$: $(x + y) + z = x + (y + z)$;
$A3$: $x + x = x$;
$A4$: $(x + y) \cdot z = x \cdot z + y \cdot z$;
$A5$: $(x \cdot y) \cdot z = x \cdot z + y \cdot z$.

(2) Axioms for PAP

$M1$: $x \parallel y = (x \between y + y \between x) + x|y$;
$LM2$: $v \between y = v \cdot y$;
$LM3$: $v \cdot x \between y = v \cdot (x \parallel y)$;
$LM4$: $(x + y) \between z = x \between z + y \between z$;
$CM5$: $v|w = \gamma(v, w)$;
$CM6$: $v|(w \cdot y) = \gamma(v, w) \cdot y$;
$CM7$: $(v \cdot x)|w = \gamma(v, w) \cdot x$;
$CM8$: $(v \cdot x)|(w \cdot y) = \gamma(v, w) \cdot (x \parallel y)$;
$CM9$: $(x + y)|z = x|z + x|z$;
$CM10$: $x|(y + z) = x|y + x|z$.

(3) Axioms for ACP

$A6$: $x + \delta = x$;
$A7$: $\delta \cdot x = \delta$;
$D1$: $v \notin H\ \partial_H(v) = v$;
$D2$: $v \in H\ \partial_H(v) = \delta$;
$D3$: $\partial_H(\delta) = \delta$;
$D4$: $\partial_H(x + y) = \partial_H(x) + \partial_H(y)$;
$D5$: $\partial_H(x \cdot y) = \partial_H(x) \cdot \partial_H(y)$;
$LM11$: $\delta \between x = \delta$;
$CM12$: $\delta|x = \delta$;
$CM13$: $x|\delta = \delta$.

(4) Recursive Definition and Specification Principles

$RDP\ \langle X_i \mid E\rangle = (\langle X_1 \mid E\rangle, \cdots, \langle X_n \mid E\rangle)(i \in \{1, \cdots, n\})$;
$RSP\ If\ y_i = t_i(y_1 \cdots y_n)(i \in \{1, \cdots, n\}), then$
$y_i = \langle X_i \mid E\rangle (i \in \{1, \cdots, n\})$.

(5) Axioms for Projection Operators

$PR1$: $\pi_n(x + y) = \pi_n(x) + \pi_n(y)$;
$PR2$: $\pi_{n+1}(v) = v$;
$PR3$: $\pi_{n+1}(v \cdot x) = v \cdot \pi_n(x)$;
$PR4$: $\pi_0(x) = \delta$;
$PR5$: $\pi_n(\delta) = \delta$.

(6) Approximation Induction Principle

$AIP\ If\ \pi_n(x) = \pi_n(y)\ for\ n \in \mathbb{N}, then\ x = y$.

(7) Axioms for the Silent Step

$B1$: $v \cdot \tau = \tau$;
$B2$: $v \cdot (\tau \cdot (x + y) + x) = v \cdot (x + y)$.

(8) Axioms for ACP τ

$TI1$: $v \notin I\ \tau_I(v) = v$;
$TI2$: $v \in I\ \tau_I(v) = \tau$;
$TI3$: $\tau_I(\delta) = \delta$;
$TI4$: $\tau_I(x + y) = \tau_I(x) + \tau_I(y)$;
$TI5$: $\tau_I(x \cdot y) = \tau_I(x) \cdot \tau_I(y)$.

(9) Cluster Fair Abstraction Rule

CFAR If X is in a cluster for I with exits
$\{v_1 Y_1, \cdots, v_m Y_m, w_1, \cdots, w_n\}, then$
$\tau \cdot \tau_I(\langle X \mid E\rangle) = \tau \cdot \tau_I(v_1\langle Y_1 \mid E\rangle + \cdots + v_m\langle Y_m \mid E\rangle + w_1 + \cdots + w_n)$.

(10) Standard Concurrency and Handshaking

$SC1$: $x \between (y||z) = (x \between y) \between z$;
$SC2$: $x \between \delta = x\delta$;
$SC3$: $x|y = y|x$;
$SC4$: $(x|y)|z = x|(y|z)$;
$SC5$ $(x|y) \between z = x|(y \between z)$;
$Handshaking x|y|z = \delta$.

B.

sort $Bit = struct\ e0|e1$;
sort $D = struct\ d0$;
sort $Ind = struct\ I\_ok|I\_nok$;
sort $Tcomm = struct\ to$;
map
inv: $Bit -> Bit$;
var

d: D;

$b0, b1$: Bit;

eqn

$inv(e0) = e1$;

$inv(e1) = e0$;

act

$r1, s4$: D;

$r2, s2, c2, s4, s3, r3, c3$: $D\#Bit\#Nat\#Nat$;

$r2, s2, c2, s4, r3$: $D\#Bit\#Bit\#Bit$;

$s3, r3, c3$: $D\#Bit$;

$s3, r3, c3$: Ind;

$r5, s5, c5$: Ind;

$r6, s6, c6, r7, s7, c7$: $Tcomm$;

$stop$;

proc

$X(b2 : Bit, rn : Nat, \max\_ : Nat)$

$= sum\ d : D.r1(d).Y(d, b2, 0, max\_)$;

$Y(d : D, b2 : Bit, rn : Nat, \max\_ : Nat)$

$= s2(d, b2, rn, \max\_).Z(d, b2, rn, max\_)$;

$Z(d : D, b2 : Bit, rn : Nat, \max\_ : Nat)$

$= r5(I\_ok).X(inv(b2), rn, max\_)$

$+r5(I\_nok).((rn == max\_)$

$- > r6(to).s7(to).stop.delta$

$< > Y(d, b2, rn + 1, max\_))$;

$V = sum\ d : D, b2 : Bit, rn : Nat, \max\_ :$
$Nat.r2(d, b2, rn, \max\_).$

$W(d, b2, rn, max\_)$;

$W(d : D, b2 : Bit, rn : Nat, \max\_ : Nat) =$
$s3(d, b2, rn, \max\_).V$

$+s3(I\_nok).((rn == max\_)$

$- > s6(to).V < > V)$;

$R = sumd : D, b2 : Bit, rn : Nat, \max\_ :$
$Nat.r3(d, b2, rn, \max\_).$

$T(d, b2, rn, max\_) + r3(I\_nok).U$;

$T(d : D, b2 : Bit, rn : Nat, \max\_ : Nat) =$
$s4(d).s5(I\_ok).R$;

$U = s5(I\_nok).(R + r7(to).R)$;

init

$\boldsymbol{allow}(\{c2, c3, c5, r1, s4, stop, c6, c7\},$

$\boldsymbol{comm}(\{r2|s2- > c2, r3|s3- > c3, r5|s5- > c5, r6|s6- >$
$c6, r7s7- > c7\},$

$X(e0, 0, 1)||V||R))$;

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The project is supported by Natural Science Foundation of the Science and Technology Department of Shaanxi Province (Grant no. 2016JM5068), Natural Science Foundation of the Key Laboratory of Shaanxi Provincial Department of Education (Grant no. 15JS075), and Natural Science Basic Research Plan in Shaanxi Province of China (Grant no. 2017JM1028). The project is also supported by Shaanxi Collaborative Innovation Center of Green Intelligent Printing and Packaging.

References

[1] A. Andersen, M. Bejlegaard, T. D. Brunoe, and K. Nielsen, "Investigating the Impact of Product Volume and Variety on Production Ramp-Up," in *Managing Complexity*, Springer Proceedings in Business and Economics, pp. 421–434, Springer International Publishing, 2017.

[2] Y. Koren, "General RMS Characteristics. Comparison with Dedicated and Flexible Systems," *Reconfigurable Manufacturing Systems and Transformable Factories*, pp. 27–45, 2006.

[3] G. Zhang, R. Liu, L. Gong, and Q. Huang, "An analytical comparison on cost and performance among DMS, AMS, FMS and RMS," in *Reconfigurable manufacturing systems and transformable factories*, A. I. Dashcenko, Ed., pp. 659–673, Springer, 2006.

[4] M. G. Mehrabi, A. G. Ulsoy, Y. Koren, and P. Heytler, "Trends and perspectives in flexible and reconfigurable manufacturing systems," *Journal of Intelligent Manufacturing*, vol. 13, no. 2, pp. 135–146, 2002.

[5] Y. Koren, U. Heisel, F. Jovane, and etal., "Reconfigurable Manufacturing Systems," *Journal of Manufacturing Systems*, vol. 29, no. 4, pp. 130–141, 1999.

[6] A. Andersen, K. Nielsen, and T. D. Brunoe, "Prerequisites and Barriers for the Development of Reconfigurable Manufacturing Systems for High Speed Ramp-up," *Procedia CIRP*, vol. 51, pp. 7–12, 2016.

[7] Z. M. Bi, S. Y. T. Lang, W. Shen, and L. Wang, "Reconfigurable manufacturing systems: the state of the art," *International Journal of Production Research*, vol. 46, no. 4, pp. 967–992, 2008.

[8] Y. Koren and M. Shpitalni, "Design of reconfigurable manufacturing systems," *Journal of Manufacturing Systems*, vol. 29, no. 4, pp. 130–141, 2010.

[9] P. Chang, Y. Lin, and J. C. Chen, "System reliability for a multi-state manufacturing network with joint buffer stations," *Journal of Manufacturing Systems*, vol. 42, pp. 170–178, 2017.

[10] D. Prasad and C. Jayswal S, "Design of Reconfigurable Manufacturing System," *National Conference on Futuristics in Mechanical Engineering*, 2017.

[11] I. Batchkova, G. Popov, and G. Stambolov, "Modeling of communications in holonic reconfigurable manufacturing system using UML," in *International Conference Amtech 2005 "advanced Manufacturing Technologies"*, vol. 44, pp. 459–465, University of Russe, Bulgaria, 2017.

[12] S. Borgo, A. Cesta, and A. Orlandini, "A Planning-based Architecture for a Reconfigurable Manufacturing System," *International Conference on Automated Planning and Scheduling*, 2016.

[13] L. Liu, J. Wenyou, and X. Dezhang, "Improved reconfigurable conveyor line based on profile joint," in *Mechanical Design and Research*, vol. 30, pp. 94–96, 103 edition, 2014.

[14] A. M. Youssef and H. A. ElMaraghy, "Optimal configuration selection for reconfigurable manufacturing systems," *International Journal of Flexible Manufacturing Systems*, vol. 19, no. 2, pp. 67–106, 2007.

[15] A. M. A. Youssef and H. A. Elmaraghy, "Modelling and optimization of multiple-aspect RMS configurations," *International Journal of Production Research*, vol. 44, no. 22, pp. 4929–4958, 2006.

[16] A. Gola and A. Świć, "Simulation Based Analysis of Reconfigurable Manufacturing System Configurations," *Applied Mechanics & Materials*, vol. 844, pp. 50–59, 2016.

[17] P. Ryan and S. Schneider, "Process algebra and non-interference," *Journal of Computer Security*, vol. 9, no. 1-2, pp. 75–103, 2001.

[18] W. Fokkink, "Basic Process Algebra," in *Introduction to Process Algebra*, Texts in Theoretical Computer Science. An EATCS Series, pp. 7–16, Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.

[19] W. Wang, Q. Zheng, and Y. Cao, "Context-Awareness Model Based on Process-Algebra," *Journal of Xian Jiao*, pp. 39-10, 2005.

[20] R. Milner, *A Calculus of Communicating Systems*, Springer, Berlin, 1980.

[21] Y. Wei, C. Lin, and Q. Tian L, "Represent and Analyze with Adaptive Workflow by Process Algebra," *Acta Electronica Sinica*, vol. 30, no. 11, pp. 1624–1628, 2002.

[22] J. F. Groote and E. P. de Vink, "Problem Solving Using Process Algebra Considered Insightful," in *ModelEd, TestEd, TrustEd*, vol. 10500 of *Lecture Notes in Computer Science*, pp. 48–63, Springer International Publishing, 2017.

[23] E. Brinksma, "A Stochastic Causality-Based Process Algebra," *The Computer Journal*, vol. 38, no. 7, pp. 552–565, 1995.

[24] J. Bergstra A and W. Klop J, "Process algebra for synchronous communication," *Information &Control*, vol. 60, no. 1, pp. 109–137, 1984.

[25] E. Scott, J. Nicol, J. Coulter, A. Hoyle, and C. Shankland, "Process Algebra with Layers: Multi-scale Integration Modelling Applied to Cancer Therapy," in *Computational Intelligence Methods for Bioinformatics and Biostatistics*, vol. 10477 of *Lecture Notes in Computer Science*, pp. 118–133, Springer International Publishing, 2016.

[26] J. Groote F, A. Mathijssen, and M. Reniers, "The Formal Specification Language mCRL2," in *Proceedings of the Dagstuhl Seminar*, 15 pages, 2007.

[27] http://www.mcrl2.org.