卒業論文

マッチング処理制約のあるフレキシブルフローショップの性能評価に関する考察

指導教員　　広島大学 工学部 第二類 (電気・電子・システム・情報系)

土肥 正　　教授

岡村 寛之　　准教授

広島大学 工学部第二類

システム工学課程

B132075

森原　和也

平成 29 年 2 月 17 日

Abstract

In this paper, we evaluate the performance of a 4 step Flexible Flow Show (FFS) with match processing constraint under a condition of defective parts occuring. FFS with occurence of defective part is modeled by queuing network and blocking probability and throughput are obtained by analysis using Markov chain. In particular, we investigate the influence of match processing constraint and inspection timing of defective parts on their performance measure.

謝辞

本研究を進めるにあたり、ご指導を頂いた卒業論文指導教員の土肥正教授，岡村寛之准教授に対し深く感謝致します．特に内部の細部に渡り，多くの貴重な助言をいただいた岡村寛之准教授に対し深く感謝致します．また、日常の議論を通じて多くの知識や示唆を頂いたディペンダブルシステム論研究室の先輩，同輩に対し深く感謝致します．

Contents

Chapter 1

Introduction

The production system which arrange machines in order to process products, is called flow shop. Among flow shop, the flo system which arrange machines parallel in each process is called Flexible Flow Shop (FFS). Parallel processing in done in FFS so the evaluation of system performance becomes importatn in design and planning phase. In past, performance evaluation on 4 step FFS with match processing constraint is done by Zhang et al. [1]. Match processing constraint is constraint which requires specific pairs when processing two different parts, and it can be bottleneck for FFS. Concretely, Zhange et al. [1] used analytical approximation method to evaluate performance of throughput and staying time in system. But in reference [1], the case when defective parts occurs were not considered and how occurence and discarding of defective parts affect FFS with match processing constraints. In this paper, we assume that in each process of 4 step FFS Zhange [1] failure can occur, and investigate the impact of occurence and removal of defective parts through perfomarnce evaluation using probablistic models. To be more specific, we will (i) compare performance for FFS with and without match processing constraint, (ii) compare the removal timing of defective parts and investigate the influence of the occurence of defective parts.

The structure of this paper is as follows. In chapter 2, we will introduce queueing network of 4 step FFS with match processing constraint which Zhang [1] used and analysis method used in this paper. In chapter 3, we clarify the relationship between match processing constraint and occurence of defective parts by performance evaluation for FFS with and without constraint under situation where defective parts can occur. In chapter 4, we consider 4 removal timings in the FFS with match processing constraints and consider the optimum removal timing by comparing their performance. In chapter 5, we state summary of this paper and future works to be done.

Chapter 2

4 step FFS with match processing constraint

2.1 4 step FFS

We think about 4 step FFS with match processing constraint considered in bibitem [1]. Figure 2.1 shows the processs in each step and relationship between steps in 4 step FFS. In the first step, two types of parts arrive separately and the parts are processed on independent lines and handed to the next step. The second step is called a matching step and in this step, two parts are combined once and processed. After processed, two parts are disassembled and then passed to next step. Here, the two parts combined once in step 2 will be pairs. In the third step, the two parts decomposed in the second step are processed in parallel on separate lines. The final fourth step is called combine step. The two parts are combined in this step and shipped from system. At this time, there are constraint that two parts combined at the fourth step must be pairs processed in the second step, and this constraint is called match processing constraint. Also, in each step, a buffer for parts is prepared, and when the buffer of the next step is full, processing of the previous step would stop since parts can not be tranferred from the previous step. This is called blocking. For the FFS illustrated in figure 2.1, if there is no overtaking of parts and the order of parts is maintained, the two parts at the head of each buffer in the fourth step are always pairs processed in the second step, so the match processing constraint is satisfied.

Zhang et al. [1], represented the FFS above with the queueing network with arrival of the two kinds of parts to the first step follows the Poisson process of the arrival rate λ, and the processing at each step follows the exponential distribution with the average $\frac{1}{\mu}$. Figure 2.2 shows 4 step FFS with match processing constraint used by Zhang et al. In this figure, white circle represent the machines for processing parts, rectangles on white circles represent buffers of each machine, each

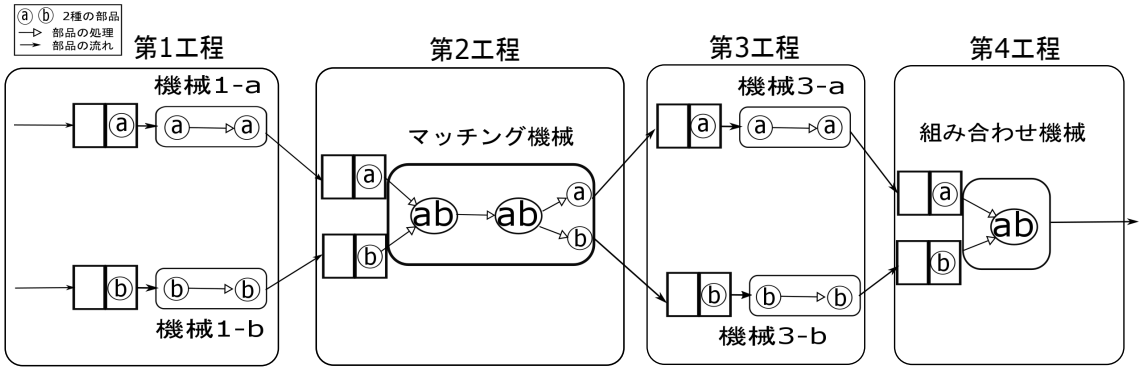

Figure 2.1: 4 step FFS with match processing constraint.

arraow represents movement of parts processed by the machine. If the arrow points to an empty place,, it means departing from the sysytem.

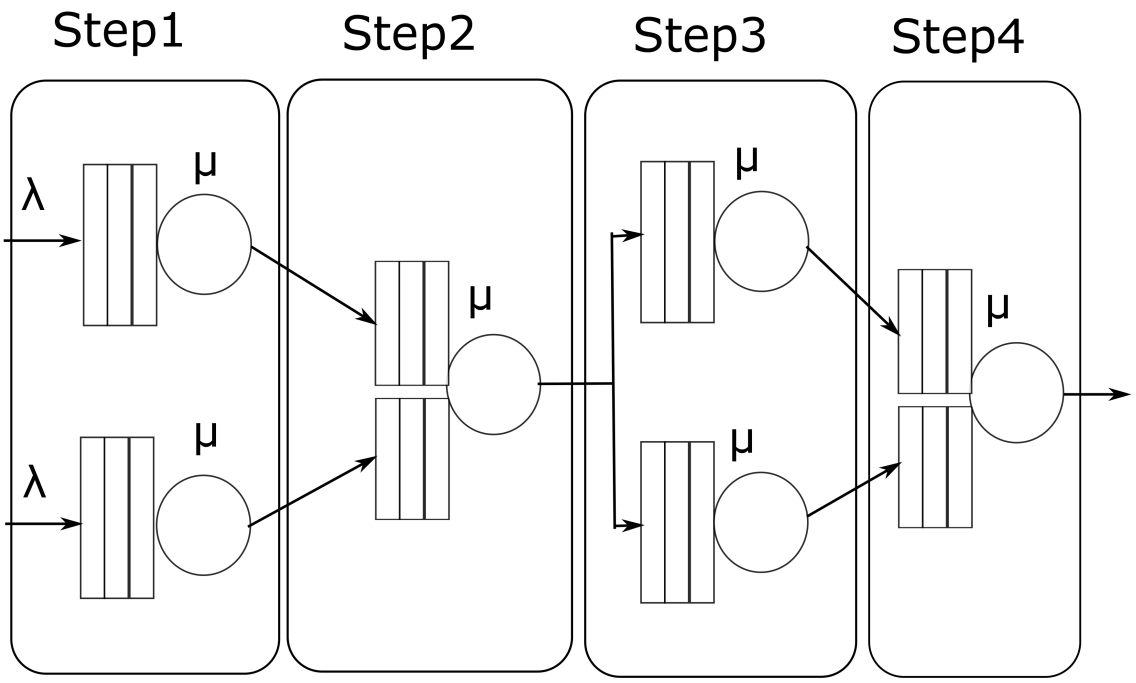

Figure 2.2: Queuing network model for 4 step FFS.

2.2 Analystic method

In this paper, we rewrite queuing network model of 4 step FFS with Generalized Stochastic Petri Net (GSPN). After that, we convert GSPN to continuous-time Markov chain (CTMC) throught GSPN analysis tool, and do steady state analysis of CTMC for performance evaluation. Analytically strict performance evaluation can be derived by this method. FIgure 2.3 shows GSPN which rewritten queuing network of figure 2.2 . For details of GSPN, see appendix A.

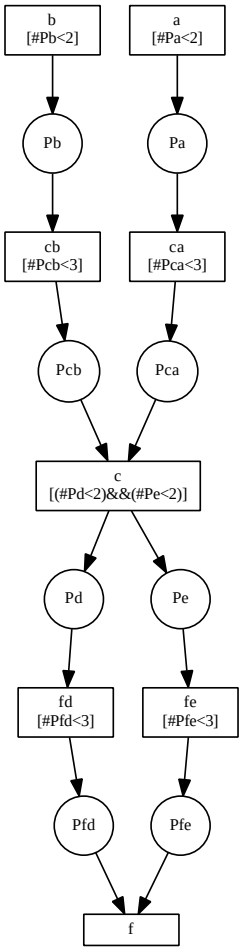

Figure 2.3: GSPN correspond for queuing network of 4 step FFS.

Chapter 3

Match processing constraint of FFS considering defective parts

3.1 FFS when failure parts occurs

In this section, we consider the situation where failure parts occurs with a certain probability in each step of 4 step FFS introduced in the previous chapter. Specifically, we set a failure probability (probability of occurence of a defective part) β per part in each step. Figure **??** shows queuing network for a FFS in consideration of occurence of defective parts. β on the arrow indicates the probability of the parts moving to the buffer arrow is pointing.

3.2 Match processing constraint when defective part occur

In FFS shown by figure 2.1, match processing constraint was always satisfied since the order of parts was always maintained. However, in the FFS which considered the removal of defective parts, shown in figure 3.1, the order of parts is not retained due to the discarding of defective parts. In this situation, we need to consider the match processing constraint in the fourth step strictly. Therefore, in this chapter, we will consider two FFSs, 4 step FFS with and without constraint to investigate the influence of match processing constraint. For FFS without constarint, we assume that regardless of the pair in the second step, conbination can be started when the two parts are gathered at fourth step. Figure 3.2 shows the GSPN for FFS with constraint and figure 3.3 shows GSPN for FFS without constraint. In this chapter, we assumed that defective parts generated in the first, second, and fourth step are immediately removed and defective parts generated in the third step, which is step related to match processing constraint, are removed before combination

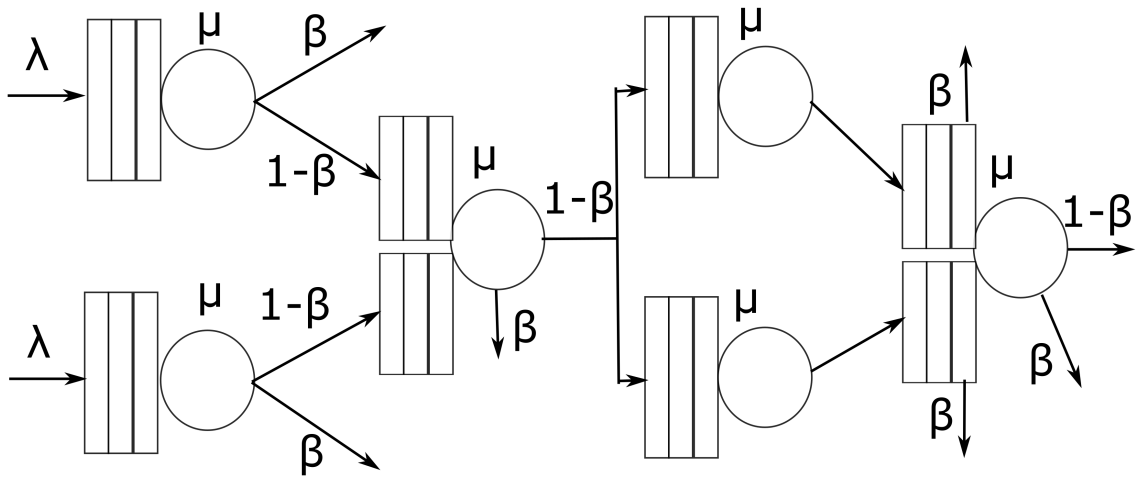

Figure 3.1: Queuing network model of FFS with removal or defective parts.

in the fourth step. This assumption also applies to FFS without constraint.

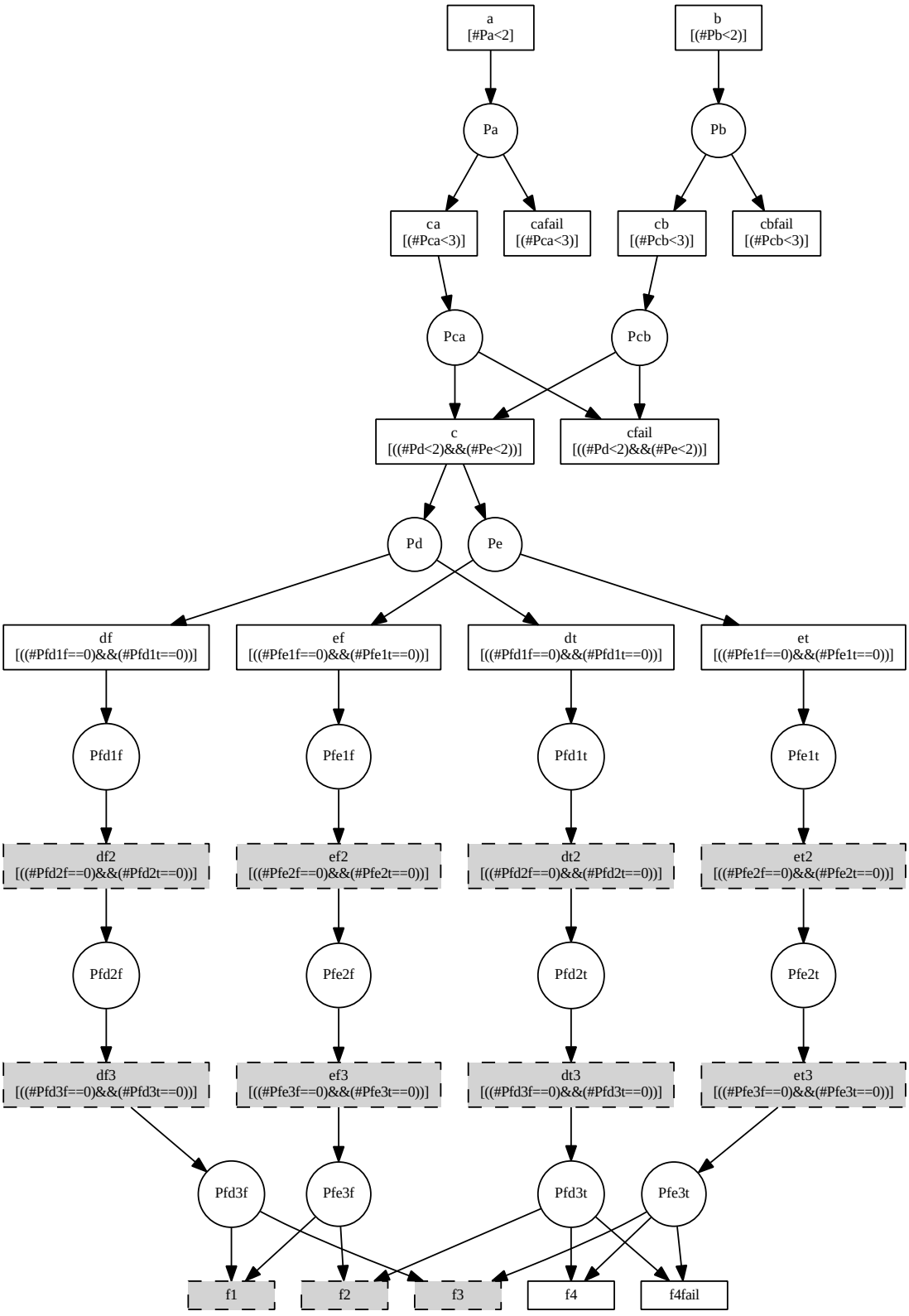

Figure 3.2: GSPN for 4 step FFS with constraint.

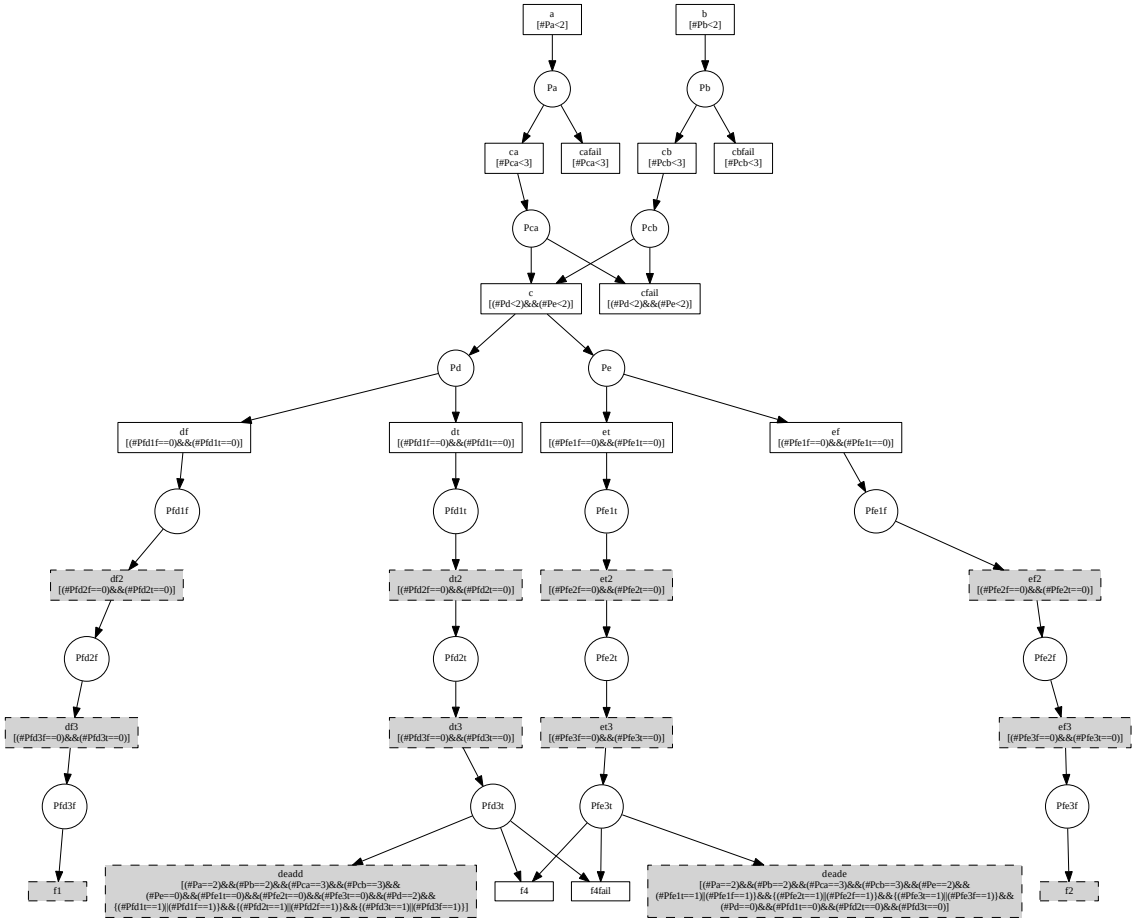

Figure 3.3: GSPN for 4 step FFS without constraint.

3.3 Numerical experiment

We are going to performance evaluation on the queuing network of two FFS mentioned in the section 3.2. We will calculate throughput and blocking probability as performance measure. Throughput is defined as the number of parts processed in the fourth step per unit time. Blocking probability calculates the probability of the buffer of fourth step to be full in steady state. In this calculation for blocking probability, since the same arrival rate and processing rate are assumed for two parts, attention is paid only to one line of the third step. Also in the experiment, we assume that size of buffer of the first and third step, which has two separte lines, are set to 1 (the part under processing is no included), and the buffer of second and fourth step, which has combination process, are set to 2 respectively.

By the GSPN analysis, if there is no match processing constraint, we found that situation which the processing of the whole step stop (deadlock) occurs when all the buffers of the first, second and one line of third, fourth step are filled, and buffer of other line of third, fourth step are empty. To avoid this, we set to remove normal parts in the fourth step when deadlock state occured.

Table 3.1 to table 3.6 shows the result of throughput (TP) and blocking probability (BP) in the third step. Experiment is done by changing the arrival rate λ and efect rate β of the two parts under the processing rate $\mu = 1.0$.

Table 3.1: Throughput and Blocking Probability（$\lambda = 0.9, \mu = 1$）.

	With constraint		Without constraint	
β	TP	BP	TP	BP
0.00	0.494	0.082	0.465	0.171
0.01	0.473	0.076	0.372	0.263
0.02	0.453	0.071	0.361	0.261
0.03	0.433	0.066	0.350	0.260
0.04	0.414	0.062	0.340	0.258
0.05	0.395	0.057	0.329	0.257

Table 3.2: Throughput and Blocking Probability（$\lambda = 0.7, \mu = 1$）.

	With constrain		Without constraint	
β	TP	BP	TP	BP
0.00	0.457	0.065	0.437	0.156
0.01	0.436	0.060	0.358	0.254
0.02	0.416	0.056	0.347	0.252
0.03	0.396	0.051	0.336	0.251
0.04	0.378	0.047	0.325	0.250
0.05	0.359	0.044	0.314	0.249

Table 3.3: Throughput and Blocking Probability（$\lambda = 0.5, \mu = 1$）.

	With constrain		Without constraint	
β	TP	BP	TP	BP
0.00	0.380	0.039	0.372	0.128
0.01	0.362	0.036	0.322	0.236
0.02	0.344	0.033	0.310	0.235
0.03	0.327	0.030	0.299	0.235
0.04	0.311	0.028	0.288	0.234
0.05	0.295	0.025	0.277	0.233

Table 3.4: Throughput and Blocking Probability ($\lambda = 0.3, \mu = 1$) .

	With constrain		Without constraint	
β	TP	BP	TP	BP
0.00	0.253	0.012	0.253	0.093
0.01	0.241	0.011	0.236	0.215
0.02	0.229	0.010	0.226	0.215
0.03	0.217	0.009	0.216	0.216
0.04	0.206	0.008	0.207	0.217
0.05	0.195	0.007	0.198	0.217

Table 3.5: Throughput and Blocking Probability ($\lambda = 0.1, \mu = 1$) .

	With constrain		Without constraint	
β	TP	BP	TP	BP
0.00	0.090	5.98E-4	0.090	0.065
0.01	0.085	5.50E-4	0.086	0.215
0.02	0.081	5.05E-4	0.082	0.216
0.03	0.077	4.63E-4	0.079	0.217
0.04	0.073	4.25E-4	0.075	0.217
0.05	0.069	3.89E-4	0.072	0.218

Table 3.6: Throughput and Blocking Probability ($\lambda = 0.01, \mu = 1$) .

	With constrain		Without constraint	
β	TP	BP	TP	BP
0.00	9.08E-3	6.54E-7	9.08E-3	0.005
0.01	8.63E-3	6.01E-7	8.70E-3	0.222
0.02	8.20E-3	5.53E-7	8.32E-3	0.222
0.03	7.79E-3	5.08E-7	7.97E-3	0.222
0.04	7.40E-3	4.66E-7	7.62E-3	0.222
0.05	7.02E-3	4.27E-7	7.29E-3	0.222

3.4 Lessons learned

From table 3.1 to 3.6 we found out that when arrival rate is higher than $\lambda = 0.3$, the FFS with match processing constraint has higher throughput, and if the arrival rate is less than $\lambda = 0.3$, the throughput of FFS without constraint got higher. We can consider that this is because when arraival rate is higher than $\lambda = 0.3$ in FFS without constraint, the blocking probability in the third step becomes very high and the arrival rate of parts arriving at the system decreases. When there is no match processing constarint, even if one of the pairs is removed in the third step, other part of the pair will wait until next part in the buffer of fourth step, and the occupancy rate of the parts in the buffer becomes relatively high. Also, as seen in the situation in which the deadlock occurs, if there is no constraint, when the buffer of one line runs out in the third step, previous step is blocked and throughput of all system decreases. Therefore, in a situation where the arrival rate is high, buffer will be occupied frequently and performance will drop when there is no constraint. On the other hand, when the arrival rate is low, there is enough buffer at third step and throuput would be higher with FFS without constraint. From these reasons, we learned that the influence of match processing constraint on throuput varies depending on the congestion degree of the system.

Chapter 4

Inspection and removal timing of defective parts

4.1 Inspection and removal timing of defective parts

The purpose of inspection and removal is to reduce occupation ratio of buffers of defective parts in each process by discarding defective parts in the middle of the system. Therefore, to evaluate the performance of FFS, the timing of inspection and removal when defective parts occurs is important. We assume that inspection is done at not all the step but in specific step. In this chapter, we consider the combination of steps that make the decrease rate of the throuput least when the number of inspection and removal of defective parts is increased from one to two. The timing of inspection and removal in each step is performed before entering the buffer of the next step, that is, immediately after the processing of each step. However, for the third step which is affected by the match processing constraint, if the defective parts occurs in one line, it is not immediately removed and sent to the buffer of the fourth step. When the pair is completed in the fourth step, if either of parts were defective one, remove. Also, defective parts that were sent to the next step without being inspected will always make defective parts in the next step. We set the size of buffer for first and third step, which has two separate line, to be 1 and second and fourth step, which has combination process, to be 2.

The place where the inspection can be performed are immediately after first, second, fourth step and after third step, which is before combination process at fourth step. As a requirement for selecting the step to be inspected, the inspection has to be done immediately after fourth step where parts finally leave the system. Therefore, if we set number of inspection to 1, the pattern of the inspection step considered from the requirement is only the one when inspection is done immediately after fourth step. If we set the number of step to be inspected to 2, the pattern of the

inspected step considered from the requirement is "immediately after the first and fourth step", "immediately after the second and fourth step", and "after the third steo and immediately after the fourth step."

Figure 4.1 shows the GSPN when inspection is done immediately after fourth step only. Figure 4.2, 4.3, 4.4 shows the GSPN corresponding to the three queuing network of the three pattern we suggested above. Since each GSPN needs to memorize the success or failure of the processing of each step, the number of states is increased more than the GSPN used in chapter **??**. We do performance evaluation using these GSPNs.

4.2 Numerical experiment

For performance measure, we calculate throughout, blocking probability, and decrease rate of throuput for four patterns of combination of steps to be inspected. The four patterns are "immediately after fourth step.", "immediately after the first and fourth step", "immediately after the second and fourth step", and "after the third steo and immediately after the fourth step." Decrease rate is a value indicating a decrease in throughput when the failure rate is changed with value when failure rate is 0% as a standard and it is obtained by following formula.

$$\text{decrease rate} = \frac{(\text{throughput when } \beta = 0) - (\text{throughput after } \beta \text{ changed})}{\text{throughput when } \beta = 0} \times 100$$

In this paper, we want to investigate the influence of increase of number of step to be inspected on throughput. Therefore, the probability of single product is processed normally (same as the decrease rate of throughput when removing defective parts immediately after fourth step) is calculated first, and using that probability as basis, when we increase the number of step to be inspected, we measure how decrease rate is improved. The table 4.1 shows the basis decrease rate.

The throughput is defined as the number of parts processed in the fourth steps per unit time. Blocking probability calculates the probability of the buffer of each step to be full in steady state. In this calculation of blocking probability, since the same arrival rate and processing rate are assumed for the two parts, attention is paid only to one line of the first and third step.

We set processing rate of each process to $\mu = 1.0$, arrival rate of parts to $\lambda = 0.7$ and $\lambda = 0.3$. Under the above conditions, we change failure rate β and calculate throughput, blocking probability, and decrease rate for all four pattern. The table 4.2 to 4.9 shows the result of experiment. The blocking probability in each step is expressed as n-BP (n = 1, 2, 3, 4) and decrease rate is expressed as DR.

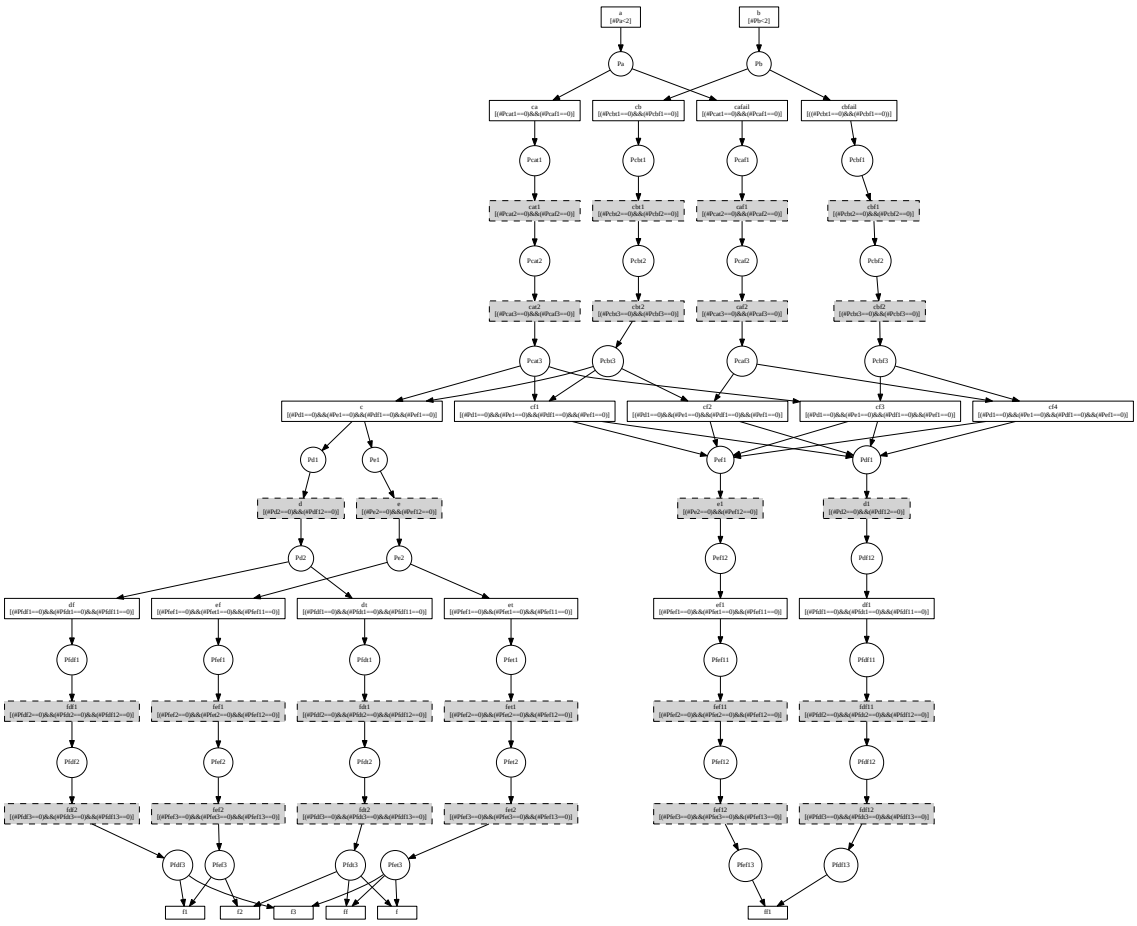

Figure 4.1: GSPN for 4 step FFS with inspection done immediately after fourth step.

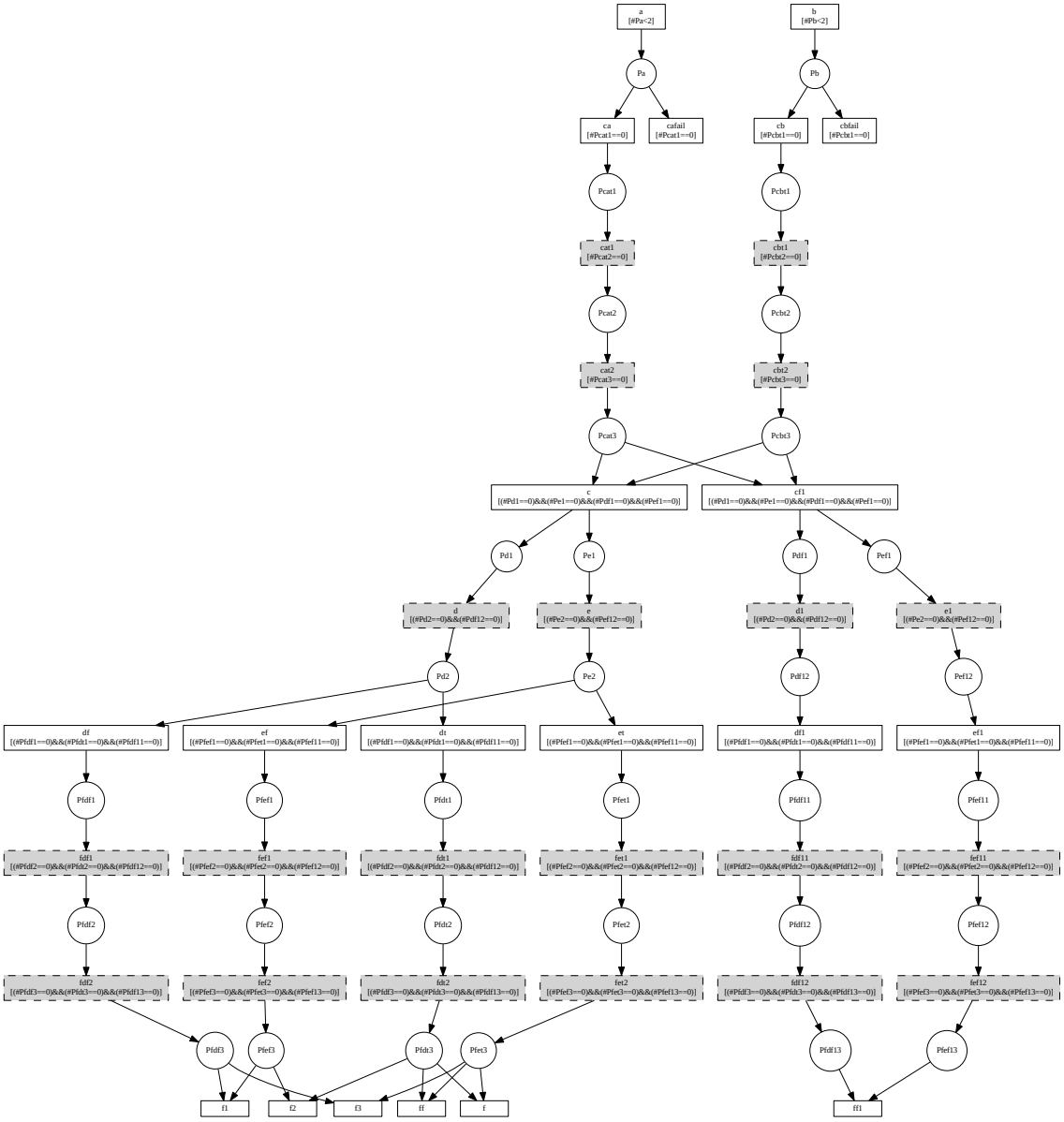

Figure 4.2: GSPN for 4 step FFS with inspection done immediately after first and fourth step.

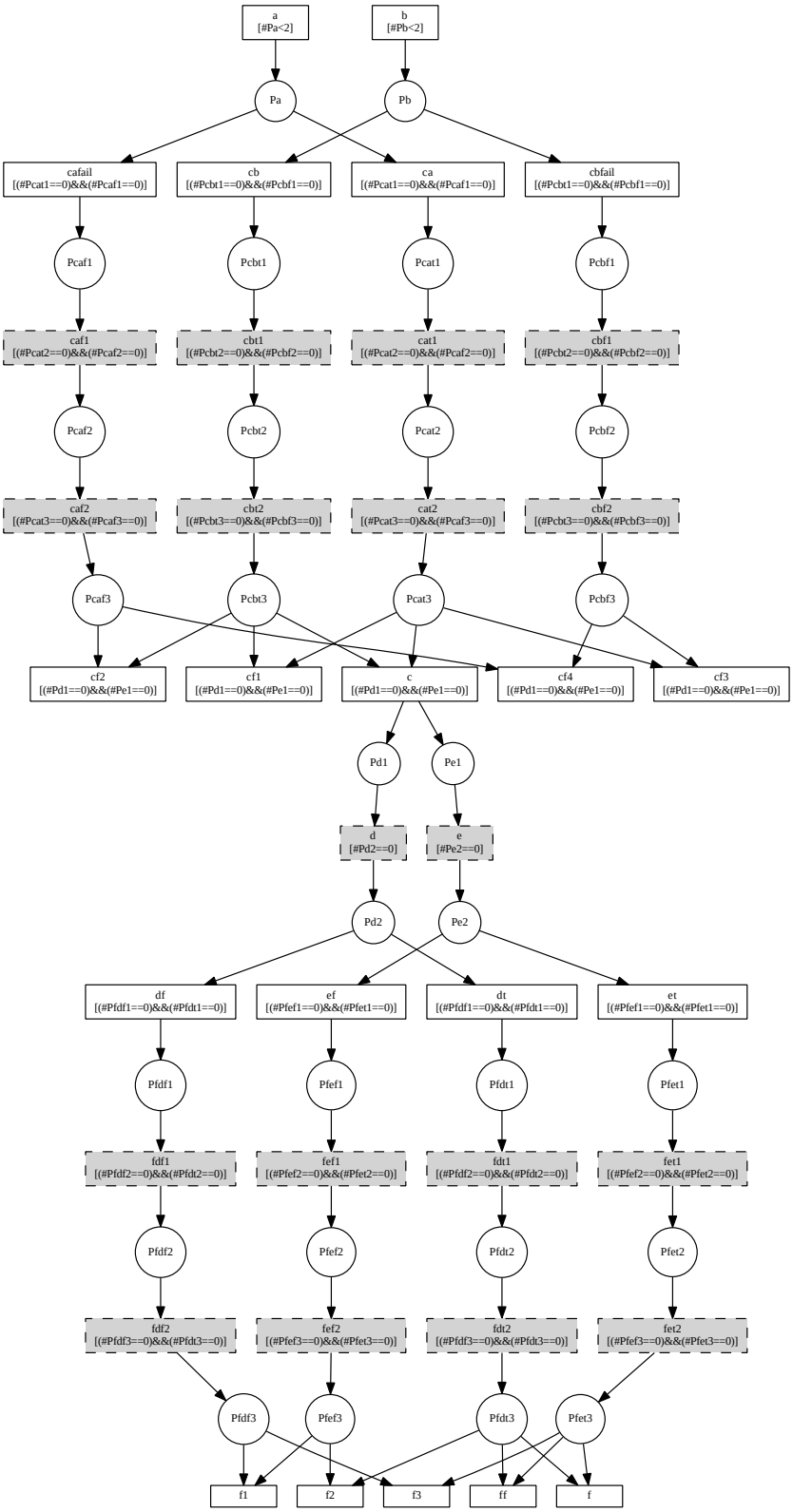

Figure 4.3: GSPN for 4 step FFS with inspection done immediately after second and fourth step.

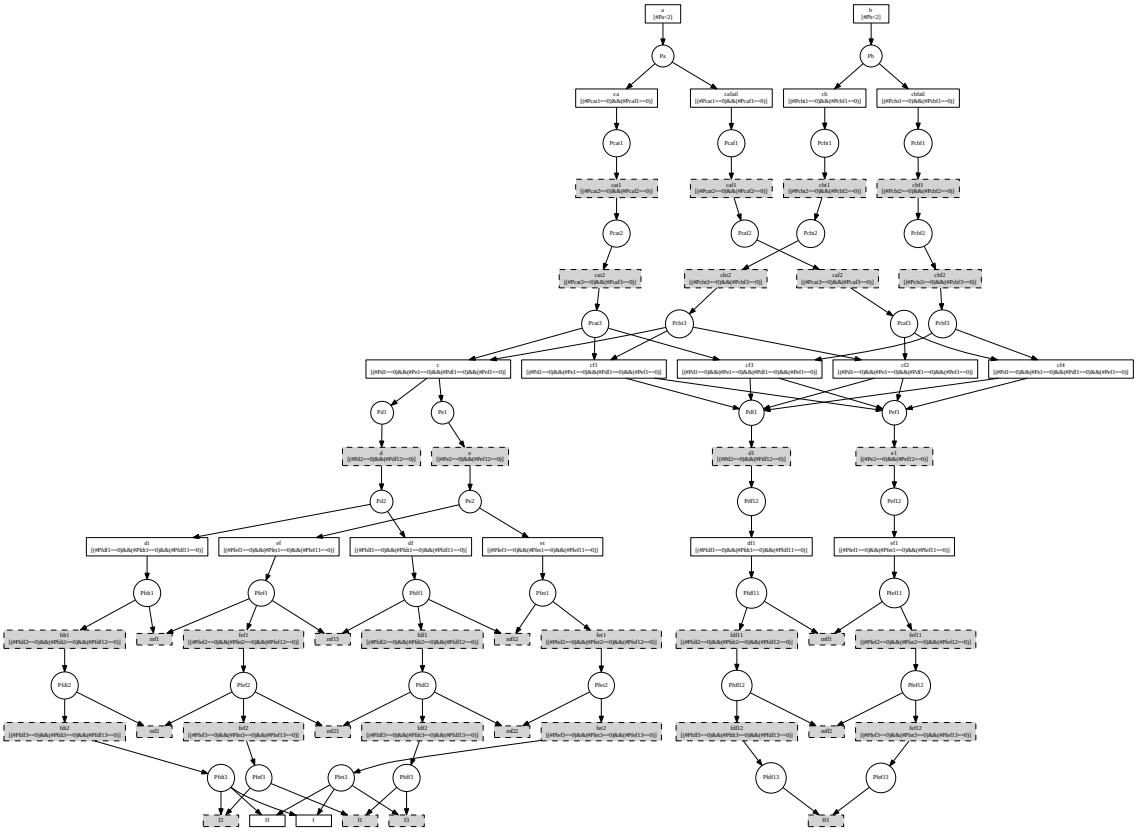

Figure 4.4: GSPN for 4 step FFS with inspection done after third step and immediately after fourth step.

Table 4.1: Probability that parts leaving the system is defective

β	decrease rate(%)
0	0
0.0001	0.060
0.01	5.85
0.02	11.4
0.03	16.7
0.04	21.7
0.05	26.5
0.1	46.9
0.5	98.4

Table 4.2: Inspection done immediately after fourth step, $\lambda = 0.7, \mu = 1.0$

β	TP	1-BP	2-BP	3-BP	4-BP	DR(%)
0.00	0.457	0.346	0.297	0.285	0.065	0
0.0001	0.456	0.346	0.297	0.285	0.065	0.060
0.01	0.430	0.346	0.297	0.285	0.065	5.85
0.02	0.404	0.346	0.297	0.285	0.065	11.4
0.03	0.381	0.346	0.297	0.285	0.065	16.7
0.04	0.357	0.346	0.297	0.285	0.065	21.7
0.05	0.336	0.346	0.297	0.285	0.065	26.5
0.1	0.243	0.346	0.297	0.285	0.065	46.9
0.5	0.007	0.346	0.297	0.285	0.065	98.4

Table 4.3: Inspection done immediately after fourth step, $\lambda = 0.3, \mu = 1.0$

β	TP	1-BP	2-BP	3-BP	4-BP	DR(%)
0.00	0.254	0.155	0.228	0.088	0.012	0
0.0001	0.253	0.155	0.228	0.088	0.01	0.060
0.01	0.239	0.155	0.228	0.088	0.012	5.85
0.02	0.225	0.155	0.228	0.088	0.012	11.4
0.03	0.211	0.155	0.228	0.088	0.012	16.7
0.04	0.198	0.155	0.228	0.088	0.012	21.7
0.05	0.186	0.155	0.228	0.088	0.012	26.5
0.1	0.135	0.155	0.228	0.088	0.012	46.9
0.5	0.003	0.155	0.228	0.088	0.012	98.4

Table 4.4: Inspection done immediately after first and fourth step, $\lambda = 0.7, \mu = 1.0$

β	TP	1-BP	2-BP	3-BP	4-BP	DR (%)
0.00	0.457	0.243	0.297	0.285	0.065	0
0.0001	0.457	0.243	0.297	0.285	0.065	0.048
0.01	0.435	0.242	0.292	0.280	0.064	4.67
0.02	0.415	0.241	0.288	0.276	0.063	9.17
0.03	0.395	0.240	0.293	0.271	0.061	13.5
0.04	0.376	0.239	0.279	0.266	0.060	17.7
0.05	0.358	0.238	0.275	0.261	0.058	21.7
0.1	0.276	0.233	0.256	0.237	0.051	39.7
0.5	0.015	0.223	0.176	0.073	0.009	96.7

Table 4.5: Inspection done immediately after first and fourth step, $\lambda = 0.3\mu = 1.0$

β	TP	1-BP	2-BP	3-BP	4-BP	DR (%)
0.00	0.254	0.046	0.228	0.088	0.012	0
0.0001	0.253	0.046	0.228	0.088	0.012	0.050
0.01	0.241	0.047	0.227	0.086	0.012	4.93
0.02	0.229	0.047	0.226	0.084	0.012	9.66
0.03	0.218	0.047	0.225	0.072	0.011	14.2
0.04	0.206	0.047	0.225	0.080	0.011	18.6
0.05	0.196	0.047	0.224	0.079	0.010	22.7
0.1	0.149	0.047	0.219	0.070	0.008	41.1
0.5	0.008	0.051	0.192	0.020	0.001	96.9

Table 4.6: Inspection done immediately after second and fourth step, $\lambda = 0.7\mu = 1.0$

β	TP	1-BP	2-BP	3-BP	4-BP	DR(%)
0.00	0.457	0.243	0.297	0.285	0.065	0
0.0001	0.456	0.243	0.296	0.285	0.065	0.056
0.01	0.432	0.241	0.291	0.269	0.060	5.48
0.02	0.408	0.239	0.285	0.254	0.056	10.7
0.03	0.385	0.238	0.280	0.239	0.052	15.8
0.04	0.363	0.236	0.276	0.225	0.047	20.6
0.05	0.342	0.235	0.272	0.212	0.044	25.3
0.1	0.250	0.230	0.255	0.155	0.028	45.4
0.5	0.007	0.221	0.221	0.005	1.62E-4	98.4

Table 4.7: Inspection done immediately after second and fourth step, $\lambda = 0.3\mu = 1.0$

β	TP	1-BP	2-BP	3-BP	4-BP	DR(%)
0.00	0.254	0.047	0.228	0.088	0.012	0
0.0001	0.253	0.046	0.228	0.088	0.012	0.060
0.01	0.239	0.046	0.227	0.083	0.011	5.84
0.02	0.225	0.046	0.227	0.078	0.010	11.4
0.03	0.211	0.046	0.227	0.073	0.009	16.7
0.04	0.199	0.046	0.226	0.069	0.009	21.7
0.05	0.186	0.046	0.226	0.065	0.008	26.5
0.1	0.135	0.046	0.225	0.047	0.005	46.8
0.5	0.004	0.046	0.222	0.001	2.55E-5	98.4

Table 4.8: Inspection done after third step and immediately after fourth step, $\lambda = 0.7, \mu = 1.0$

β	TP	1-BP	2-BP	3-BP	4-BP	DR(%)
0.00	0.457	0.347	0.297	0.285	0.065	0
0.0001	0.457	0.347	0.297	0.285	0.0065	0.059
0.01	0.431	0.346	0.296	0.284	0.065	5.77
0.02	0.406	0.346	0.295	0.282	0.066	11.3
0.03	0.382	0.346	0.295	0.281	0.068	16.5
0.04	0.359	0.346	0.294	0.280	0.070	21.5
0.05	0.337	0.345	0.294	0.279	0.072	26.2
0.1	0.244	0.345	0.294	0.279	0.089	46.6
0.5	0.007	0.350	0.303	0.298	0.206	98.4

Table 4.9: Inspection done after third step and immediately after fourth step, $\lambda = 0.3\mu = 1.0$

β	TP	1-BP	2-BP	3-BP	4-BP	DR(%)
0.00	0.254	0.155	0.228	0.088	0.012	0
0.0001	0.253	0.155	0.228	0.088	0.012	0.060
0.01	0.239	0.155	0.228	0.088	0.015	5.85
0.02	0.225	0.155	0.228	0.088	0.018	11.4
0.03	0.211	0.155	0.228	0.087	0.021	16.7
0.04	0.198	0.155	0.228	0.087	0.024	21.7
0.05	0.186	0.155	0.228	0.087	0.026	26.5
0.1	0.135	0.155	0.228	0.087	0.041	46.9
0.5	0.004	0.155	0.228	0.086	0.113	98.4

4.3 Lessons learned

From table 4.4 to 4.9, we can see that when number of inspected step increased from one to two, decrease rate in all pattern is lower compared to the decrease rate of the basis table 4.1. Thisi is considered to be because the blocking probability is lowered by making vacancies in the buffer by removing defective parts in the middle of the system. Also, among the three patterns of having two inspection step, the decrease rate is the smallest when the inspction was set immediately after first and fourth step. We can consider the reason for this is because of the following assumption, defective parts sent to next step will make parts defective in later step. When inspection is done immediately after first and fourth step, when a defective part occurs in one line of the first step, there is no influence on the other line, and only one defective part is removed. In contrast, after second step where combination and match processing constraint affect, even if one of the part sent from first step is normal, if the other part of the pair is a defective part, a normal part is also removed together with defective part. Since many normal parts are discarded in this way, it can be considered that the decrease rate of the throughput is small in the other two pattern.

For the lessons, we learned that increasing the number of inspection steps reduces the decrease rate in throuput and throughput will improve. Moreover, when the number of inspection steps in this 4 step FFS was two, we can obtain that when defective parts are removed from the FFS immediately after first and fourth step, the throughput becomes the highest.

4.4 Summary

In this paper, we investigated and examined the influence on the performance of FFS by the match processing constraint and the number of inspection step of defective parts under the condition that defective parts occur in 4 step FFS. As a result, the influence of the match processing constraint varies depending on the congestion degree of buffer of the FFS, and as the inspection removal timing of defective parts is carried out "immediately after the first and fourth step," the lowering of the throughput is suppressed.

The FFS model we used in this research assumes a situation where the blocking probability is relatively high and the throughput is low because of small buffer size. For the future work, we will change the buffer size to a larger one and investigate whether the buffer size affects the throughput. We will also investigate the impact of defective parts on various production systems other than FFS.

Appendix A

Detail of Generalized Stochastic Petri Net (GSPN)

In this chapter, we will explain the generalized Stochastic Petri Net (GSPN) used in the main text. First, Petri Net (PN) is expressed as a directed bipartite graph with two kinds of nodes called places and transitions, and it is used as a modeling language for modeling discrete events efficiently. The PN consists of 3 tuples of places (open circles), transitions (rectangles), and tokens (filled circles). The tokens in the place move as the transition fires. In the PN, the behavior of the system is described by the state of the token (marking) for all places. When a token exists in the place for the input of the trasition like t1 of figure A.1, the trasition's state is called "can fire", and by firing, the token in the input place is moved to output place. The PN after the t1 fired is shown at figure A.2. Modeling of various systems can be done by setting the timing for when for when the trasition fires from when it get to the state "can fire" and by setting the ignitable marking.

PN is categorized according to transition firing timing. In general, PNs composed of transitions whose firing delay follow the exponential distribution are called Stochastic Petri Net (SPN), and among them, those that allow instant transitions partially are called Generalized Stochastic Petri Net (GSPN).

In the PN diagram of this paper, places are represented by white circles, exponential transitions are represented by rectangles with solid lines, and instant transitions are represented by rectangles with dotted lines. As a method of analysis, we created a matrix for generating, grouping and analyszeing Petri Net marking using PN analysis tool developed at Hiroshima University called JSPetriNet [2]. Then using Ruby's script, we convert the matrix to the transition matrix of continuous-time Markov chain (CTMC) that can be used in R. At last, we did steady state analysis

of CTMC using R.

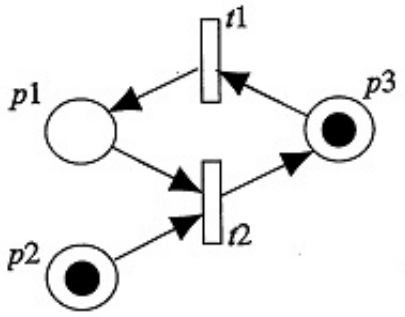

Figure A.1: Petri Net with t1 "can fire"

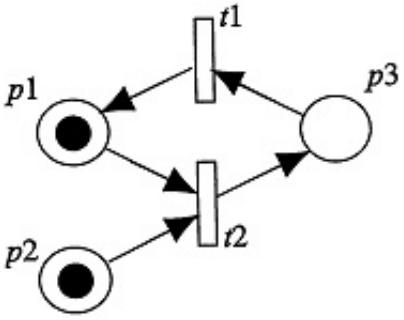

Figure A.2: Petri Net after t1 fired

Appendix B

Source code of GSPN

Here we introduce the source code used for Petri Net generation in JSPetriNet, corresponding to thePetri Net introduced in the main text.

B.1 Source code of figure 2.3

```
place Pa
place Pb
place Pca
place Pcb
place Pd
place Pe
place Pfd
place Pfe


exp a (guard = ga)
exp b (guard = gb)
exp ca (guard = gca)
exp cb (guard = gcb)
exp c (guard = gc)
exp fd (guard =gfd)
exp fe (guard = gfe)
exp f


arc a to Pa
arc b to Pb
arc Pa to ca
arc Pb to cb
arc ca to Pca
arc cb to Pcb
arc Pca to c
arc Pcb to c
```

```
arc c to Pd
arc c to Pe
arc Pd to fd
arc Pe to fe
arc fd to Pfd
arc fe to Pfe
arc Pfd to f
arc Pfe to f


na = 2
nb = 2
nca = 3
ncb = 3
nd = 2
ne = 2
nfd = 3
nfe = 3

ga := #Pa < na
gb := #Pb < nb
gca := #Pca < nca
gcb := #Pcb < ncb
gc := #Pd < nd && #Pe < ne
gfd := #Pfd < nfd
gfe := #Pfe < nfe

rwd.f := ifelse(#Pfd >= 1 && #Pfe >= 1, 1, 0)
rwd.ablock := ifelse(#Pa==2, 1, 0)
rwd.cablock := ifelse(#Pca==3, 1, 0)
rwd.cblock := ifelse(#Pd==2 || #Pe==2, 1, 0)
rwd.dblock := ifelse(#Pfd==3, 1, 0)
```

B.2 Source code of figure 3.2

```
place Pa
place Pb
place Pca
place Pcb
place Pd
place Pe

place Pfd1f
place Pfd2f
place Pfd3f
place Pfd1t
place Pfd2t
place Pfd3t
place Pfe1f
place Pfe2f
place Pfe3f
place Pfe1t
place Pfe2t
place Pfe3t


exp a (guard = ga)
exp b (guard = gb)

exp ca (guard = gca)
exp cafail (guard = gca)

exp cb (guard = gcb)
exp cbfail (guard = gcb)

exp c (guard = gc)
exp cfail (guard = gc)

exp df (guard = gd1)
exp dt (guard = gd1)
imm df2 (guard = gd2)
imm dt2 (guard = gd2)
imm df3 (guard = gd3)
imm dt3 (guard = gd3)
```

```
exp ef (guard = ge1)
exp et (guard = ge1)
imm ef2 (guard = ge2)
imm et2 (guard = ge2)
imm ef3 (guard = ge3)
imm et3 (guard = ge3)

imm f1
imm f2
imm f3
exp f4
exp f4fail

arc a to Pa
arc b to Pb

arc Pa to ca
arc Pa to cafail

arc Pb to cb
arc Pb to cbfail

arc ca to Pca
arc cb to Pcb

arc Pca to c
arc Pcb to c
arc Pca to cfail
arc Pcb to cfail

arc c to Pd
arc c to Pe

arc Pd to df
arc Pd to dt

arc df to Pfd1f
arc Pfd1f to df2
arc df2 to Pfd2f
arc Pfd2f to df3
arc df3 to Pfd3f

arc dt to Pfd1t
arc Pfd1t to dt2
arc dt2 to Pfd2t
arc Pfd2t to dt3
arc dt3 to Pfd3t

arc Pe to ef
arc Pe to et
```

```
arc ef to Pfe1f
arc Pfe1f to ef2
arc ef2 to Pfe2f
arc Pfe2f to ef3
arc ef3 to Pfe3f

arc et to Pfe1t
arc Pfe1t to et2
arc et2 to Pfe2t
arc Pfe2t to et3
arc et3 to Pfe3t

arc Pfd3f to f1
arc Pfe3f to f1
arc Pfd3t to f2
arc Pfe3f to f2
arc Pfd3f to f3
arc Pfe3t to f3

arc Pfd3t to f4
arc Pfe3t to f4
arc Pfd3t to f4fail
arc Pfe3t to f4fail

ga := #Pa < 2
gb := #Pb < 2
gca := #Pca < 3
gcb := #Pcb < 3
gc := #Pd < 2 && #Pe < 2

gd1 := #Pfd1f == 0 && #Pfd1t == 0
gd2 := #Pfd2f == 0 && #Pfd2t == 0
gd3 := #Pfd3f == 0 && #Pfd3t == 0
ge1 := #Pfe1f == 0 && #Pfe1t == 0
ge2 := #Pfe2f == 0 && #Pfe2t == 0
ge3 := #Pfe3f == 0 && #Pfe3t == 0

rwd.f := ifelse(?f4, f4.rate, 0)
rwd.ablock := ifelse(#Pa == 2, 1,0)
rwd.cablock := ifelse(#Pca==3, 1, 0)
rwd.cblock := ifelse(#Pd==2 || #Pe==2, 1, 0)
rwd.dblock := ifelse(#Pfd1f == 1 || #Pfd1t == 1, 1, 0)
```

B.3 Source code of figure 3.3

```
place Pa
place Pb
place Pca
place Pcb
place Pd
place Pe

place Pfd1f
place Pfd2f
place Pfd3f
place Pfd1t
place Pfd2t
place Pfd3t
place Pfe1f
place Pfe2f
place Pfe3f
place Pfe1t
place Pfe2t
place Pfe3t

exp a (guard = ga)

exp b (guard = gb)

exp ca (guard = gca)
exp cafail (guard = gca)

exp cb (guard = gcb)
exp cbfail (guard = gcb)

exp c (guard = gc)
exp cfail (guard = gc)

exp df (guard = gd1)
exp dt (guard = gd1)
imm df2 (guard = gd2, priority=1)
imm dt2 (guard = gd2, priority=1)
imm df3 (guard = gd3, priority=1)
imm dt3 (guard = gd3, priority=1)
```

```
exp ef (guard = ge1)
exp et (guard = ge1)
imm ef2 (guard = ge2, priority=1)
imm et2 (guard = ge2, priority=1)
imm ef3 (guard = ge3, priority=1)
imm et3 (guard = ge3, priority=1)

imm f1
imm f2
exp f4
exp f4fail

imm deadd (guard = gdeadd)
imm deade (guard = gdeade)

arc a to Pa
arc b to Pb

arc Pa to ca
arc Pa to cafail

arc Pb to cb
arc Pb to cbfail

arc ca to Pca
arc cb to Pcb

arc Pca to c
arc Pcb to c
arc Pca to cfail
arc Pcb to cfail

arc c to Pd
arc c to Pe

arc Pd to df
arc Pd to dt

arc df to Pfd1f
arc Pfd1f to df2
arc df2 to Pfd2f
arc Pfd2f to df3
arc df3 to Pfd3f

arc dt to Pfd1t
arc Pfd1t to dt2
arc dt2 to Pfd2t
arc Pfd2t to dt3
arc dt3 to Pfd3t

arc Pe to ef
arc Pe to et

arc ef to Pfe1f
arc Pfe1f to ef2
arc ef2 to Pfe2f
arc Pfe2f to ef3
arc ef3 to Pfe3f
```

```
arc et to Pfe1t
arc Pfe1t to et2
arc et2 to Pfe2t
arc Pfe2t to et3
arc et3 to Pfe3t

arc Pfd3f to f1
arc Pfe3f to f2

arc Pfd3t to f4
arc Pfe3t to f4
arc Pfd3t to f4fail
arc Pfe3t to f4fail

arc Pfd3t to deadd
arc Pfe3t to deade

ga := #Pa < 2
gb := #Pb < 2
gca := #Pca < 3
gcb := #Pcb < 3
gc := #Pd < 2 && #Pe < 2

gd1 := #Pfd1f == 0 && #Pfd1t == 0
gd2 := #Pfd2f == 0 && #Pfd2t == 0
gd3 := #Pfd3f == 0 && #Pfd3t == 0
ge1 := #Pfe1f == 0 && #Pfe1t == 0
ge2 := #Pfe2f == 0 && #Pfe2t == 0
ge3 := #Pfe3f == 0 && #Pfe3t == 0

gdeade := #Pa == 2 && #Pb == 2 && #Pca == 3 && #Pcb == 3 && #Pe == 2 && (#Pfe1t == 1 || #Pfe1f == 1) &&
          (#Pfe2t == 1 || #Pfe2f == 1) && (#Pfe3t == 1 || #Pfe3f == 1) && #Pd == 0 && #Pfd1t == 0 &&
           #Pfd2t == 0 && #Pfd3t == 0
gdeadd := #Pa == 2 && #Pb == 2 && #Pca == 3 && #Pcb == 3 && #Pe == 0 && #Pfe1t == 0 && #Pfe2t == 0 &&
           #Pfe3t == 0 && #Pd == 2 && (#Pfd1t == 1 || #Pfd1f == 1) && (#Pfd2t == 1 || #Pfd2f == 1) &&
           (#Pfd3t == 1 || #Pfd3f == 1)

rwd.f := ifelse(?f4, f4.rate, 0)
rwd.ablock := ifelse(#Pa==2, 1,0)
rwd.cablock := ifelse(#Pca==3, 1, 0)
rwd.cblock := ifelse(#Pd==2 || #Pe==2, 1, 0)
rwd.dblock := ifelse(#Pfd1f==1 || #Pfd1t==1, 1, 0)
```

B.4 Source code of figure 4.1

```
place Pa
place Pb

place Pcat1
place Pcat2
place Pcat3

place Pcaf1
place Pcaf2
place Pcaf3

place Pcbt1
place Pcbt2
place Pcbt3

place Pcbf1
place Pcbf2
place Pcbf3

place Pd1
place Pd2

place Pe1
place Pe2

place Pdf1
place Pdf12

place Pef1
place Pef12

place Pfdt1
place Pfdt2
place Pfdt3
place Pfdf1
place Pfdf2
place Pfdf3

place Pfet1
place Pfet2
place Pfet3
```

```
place Pfef1
place Pfef2
place Pfef3

place Pfdf11
place Pfdf12
place Pfdf13
place Pfef11
place Pfef12
place Pfef13


exp a (guard = ga)
exp b (guard = gb)

exp ca (guard = gca1)
exp cafail(guard = gca1)
imm cat1 (guard = gca2)
imm cat2 (guard = gca3)
imm caf1 (guard = gca2)
imm caf2 (guard = gca3)

exp cb (guard = gcb1)
exp cbfail (guard = gcb1)
imm cbt1 (guard = gcb2)
imm cbt2 (guard = gcb3)
imm cbf1 (guard = gcb2)
imm cbf2 (guard = gcb3)

exp c (guard = gc)
exp cf1 (guard = gc)
exp cf2 (guard = gc)
exp cf3 (guard = gc)
exp cf4 (guard = gc)

imm d (guard = gd)
imm d1 (guard = gd)

imm e (guard = ge)
imm e1 (guard = ge)

exp dt (guard = gd1)
exp df (guard = gd1)
imm fdt1 (guard = gd2)
imm fdt2 (guard = gd3)
imm fdf1 (guard = gd2)
imm fdf2 (guard = gd3)

exp df1 (guard = gd1)
imm fdf11 (guard = gd2)
imm fdf12 (guard = gd3)

exp et (guard = ge1)
exp ef (guard = ge1)
imm fet1 (guard = ge2)
imm fet2 (guard = ge3)
imm fef1 (guard = ge2)
imm fef2 (guard = ge3)
```

```
exp ef1 (guard = ge1)
imm fef11 (guard = ge2)
imm fef12 (guard = ge3)


exp f
exp ff
exp f1
exp f2
exp f3
exp ff1

arc a to Pa
arc b to Pb

arc Pa to ca
arc ca to Pcat1
arc Pcat1 to cat1
arc cat1 to Pcat2
arc Pcat2 to cat2
arc cat2 to Pcat3

arc Pa to cafail
arc cafail to Pcaf1
arc Pcaf1 to caf1
arc caf1 to Pcaf2
arc Pcaf2 to caf2
arc caf2 to Pcaf3

arc Pb to cb
arc cb to Pcbt1
arc Pcbt1 to cbt1
arc cbt1 to Pcbt2
arc Pcbt2 to cbt2
arc cbt2 to Pcbt3

arc Pb to cbfail
arc cbfail to Pcbf1
arc Pcbf1 to cbf1
arc cbf1 to Pcbf2
arc Pcbf2 to cbf2
arc cbf2 to Pcbf3


arc Pcat3 to c
arc Pcbt3 to c
arc Pcat3 to cf1
arc Pcbt3 to cf1
arc Pcaf3 to cf2
arc Pcbt3 to cf2
arc Pcat3 to cf3
arc Pcbf3 to cf3
arc Pcaf3 to cf4
arc Pcbf3 to cf4

arc c to Pd1
arc c to Pe1
arc Pd1 to d
arc d to Pd2
arc Pe1 to e
arc e to Pe2
```

```
arc cf1 to Pdf1
arc cf1 to Pef1
arc Pdf1 to d1
arc d1 to Pdf12
arc Pef1 to e1
arc e1 to Pef12

arc cf2 to Pdf1
arc cf2 to Pef1

arc cf3 to Pdf1
arc cf3 to Pef1

arc cf4 to Pdf1
arc cf4 to Pef1

arc Pd2 to dt
arc Pd2 to df

arc df to Pfdf1
arc Pfdf1 to fdf1
arc fdf1 to Pfdf2
arc Pfdf2 to fdf2
arc fdf2 to Pfdf3

arc dt to Pfdt1
arc Pfdt1 to fdt1
arc fdt1 to Pfdt2
arc Pfdt2 to fdt2
arc fdt2 to Pfdt3


arc Pe2 to et
arc Pe2 to ef

arc ef to Pfef1
arc Pfef1 to fef1
arc fef1 to Pfef2
arc Pfef2 to fef2
arc fef2 to Pfef3

arc et to Pfet1
arc Pfet1 to fet1
arc fet1 to Pfet2
arc Pfet2 to fet2
arc fet2 to Pfet3


arc Pdf12 to df1
arc df1 to Pfdf11
arc Pfdf11 to fdf11
arc fdf11 to Pfdf12
arc Pfdf12 to fdf12
arc fdf12 to Pfdf13

arc Pef12 to ef1
arc ef1 to Pfef11
arc Pfef11 to fef11
arc fef11 to Pfef12
arc Pfef12 to fef12
arc fef12 to Pfef13
```

```
arc Pfdf13 to ff1
arc Pfef13 to ff1


arc Pfdf3 to f1
arc Pfef3 to f1
arc Pfdt3 to f2
arc Pfef3 to f2
arc Pfdf3 to f3
arc Pfet3 to f3

arc Pfdt3 to f
arc Pfet3 to f
arc Pfdt3 to ff
arc Pfet3 to ff


ga := #Pa < 2
gb := #Pb < 2

gca1 := #Pcat1 == 0 && #Pcaf1 == 0
gca2 := #Pcat2 == 0 && #Pcaf2 == 0
gca3 := #Pcat3 == 0 && #Pcaf3 == 0

gcb1 := #Pcbt1 == 0 && #Pcbf1 == 0
gcb2 := #Pcbt2 == 0 && #Pcbf2 == 0
gcb3 := #Pcbt3 == 0 && #Pcbf3 == 0

gc := #Pd1==0 && #Pe1==0 && #Pdf1==0 && #Pef1==0
gd := #Pd2==0 && #Pdf12==0
ge := #Pe2==0 && #Pef12==0

gd1 := #Pfdf1==0 && #Pfdt1==0 && #Pfdf11==0
gd2 := #Pfdf2==0 && #Pfdt2==0 && #Pfdf12==0
gd3 := #Pfdf3==0 && #Pfdt3==0 && #Pfdf13==0

ge1 := #Pfef1==0 && #Pfet1==0 && #Pfef11==0
ge2 := #Pfef2==0 && #Pfet2==0 && #Pfef12==0
ge3 := #Pfef3==0 && #Pfet3==0 && #Pfef13==0

rwd.f := ifelse(#Pfdt3==1 && #Pfet3==1, f.rate, 0)
rwd.ablock := ifelse(#Pa==2, 1, 0)
rwd.cablock := ifelse(#Pcat1==1 || #Pcaf1==1, 1, 0)
rwd.cblock := ifelse(#Pd1==1 || #Pe1==1 || #Pdf1==1 || #Pef1==1, 1, 0)
rwd.dblock := ifelse(#Pfdt1==1 || #Pfdf1==1 || #Pfdf11==1, 1, 0)
```

B.5 Source code of figure 4.2

```
place Pa
place Pb

place Pcat1
place Pcat2
place Pcat3

place Pcbt1
place Pcbt2
place Pcbt3

place Pd1
place Pd2

place Pe1
place Pe2

place Pdf1
place Pdf12
place Pef1
place Pef12

place Pfdt1
place Pfdt2
place Pfdt3
place Pfdf1
place Pfdf2
place Pfdf3

place Pfet1
place Pfet2
place Pfet3
place Pfef1
place Pfef2
place Pfef3

place Pfdf11
place Pfdf12
place Pfdf13
```

```
place Pfef11
place Pfef12
place Pfef13


exp a (guard = ga)
exp b (guard = gb)

exp ca (guard = gca1)
exp cafail(guard = gca1)
imm cat1 (guard = gca2)
imm cat2 (guard = gca3)

exp cb (guard = gcb1)
exp cbfail (guard = gcb1)
imm cbt1 (guard = gcb2)
imm cbt2 (guard = gcb3)

exp c (guard = gc)
exp cf1 (guard = gc)

imm d (guard = gd)
imm d1 (guard = gd)

imm e (guard = ge)
imm e1 (guard = ge)

exp dt (guard = gd1)
exp df (guard = gd1)
imm fdt1 (guard = gd2)
imm fdt2 (guard = gd3)
imm fdf1 (guard = gd2)
imm fdf2 (guard = gd3)

exp et (guard = ge1)
exp ef (guard = ge1)
imm fet1 (guard = ge2)
imm fet2 (guard = ge3)
imm fef1 (guard = ge2)
imm fef2 (guard = ge3)

exp df1 (guard = gd1)
imm fdf11 (guard = gd2)
imm fdf12 (guard = gd3)

exp ef1 (guard = ge1)
imm fef11 (guard = ge2)
imm fef12 (guard = ge3)

exp f
exp ff
exp f1
exp f2
exp f3
exp ff1


arc a to Pa
arc b to Pb
```

```
arc Pa to ca
arc ca to Pcat1
arc Pcat1 to cat1
arc cat1 to Pcat2
arc Pcat2 to cat2
arc cat2 to Pcat3

arc Pb to cb
arc cb to Pcbt1
arc Pcbt1 to cbt1
arc cbt1 to Pcbt2
arc Pcbt2 to cbt2
arc cbt2 to Pcbt3

arc Pa to cafail
arc Pb to cbfail

arc Pcat3 to c
arc Pcbt3 to c
arc Pcat3 to cf1
arc Pcbt3 to cf1

arc c to Pd1
arc c to Pe1
arc Pd1 to d
arc d to Pd2
arc Pe1 to e
arc e to Pe2

arc cf1 to Pdf1
arc cf1 to Pef1
arc Pdf1 to d1
arc d1 to Pdf12
arc Pef1 to e1
arc e1 to Pef12

arc Pd2 to dt
arc Pd2 to df

arc df to Pfdf1
arc Pfdf1 to fdf1
arc fdf1 to Pfdf2
arc Pfdf2 to fdf2
arc fdf2 to Pfdf3

arc dt to Pfdt1
arc Pfdt1 to fdt1
arc fdt1 to Pfdt2
arc Pfdt2 to fdt2
arc fdt2 to Pfdt3


arc Pe2 to et
arc Pe2 to ef

arc ef to Pfef1
arc Pfef1 to fef1
arc fef1 to Pfef2
arc Pfef2 to fef2
arc fef2 to Pfef3
```

```
arc et to Pfet1
arc Pfet1 to fet1
arc fet1 to Pfet2
arc Pfet2 to fet2
arc fet2 to Pfet3

arc Pdf12 to df1
arc df1 to Pfdf11
arc Pfdf11 to fdf11
arc fdf11 to Pfdf12
arc Pfdf12 to fdf12
arc fdf12 to Pfdf13

arc Pef12 to ef1
arc ef1 to Pfef11
arc Pfef11 to fef11
arc fef11 to Pfef12
arc Pfef12 to fef12
arc fef12 to Pfef13

arc Pfdf13 to ff1
arc Pfef13 to ff1

arc Pfdf3 to f1
arc Pfef3 to f1
arc Pfdt3 to f2
arc Pfef3 to f2
arc Pfdf3 to f3
arc Pfet3 to f3

arc Pfdt3 to f
arc Pfet3 to f
arc Pfdt3 to ff
arc Pfet3 to ff


ga := #Pa < 2
gb := #Pb < 2

gca1 := #Pcat1 == 0
gca2 := #Pcat2 == 0
gca3 := #Pcat3 == 0

gcb1 := #Pcbt1 == 0
gcb2 := #Pcbt2 == 0
gcb3 := #Pcbt3 == 0

gc := #Pd1==0 && #Pe1==0 && #Pdf1==0 && #Pef1==0
gd := #Pd2==0 && #Pdf12==0
ge := #Pe2==0 && #Pef12==0

gd1 := #Pfdf1==0 && #Pfdt1==0 && #Pfdf11==0
gd2 := #Pfdf2==0 && #Pfdt2==0 && #Pfdf12==0
gd3 := #Pfdf3==0 && #Pfdt3==0 && #Pfdf13==0

ge1 := #Pfef1==0 && #Pfet1==0 && #Pfef11==0
ge2 := #Pfef2==0 && #Pfet2==0 && #Pfef12==0
ge3 := #Pfef3==0 && #Pfet3==0 && #Pfef13==0

rwd.f := ifelse(#Pfdt3==1 && #Pfet3==1, f.rate, 0)
rwd.ablock := ifelse(#Pa == 2, a.rate, 0)
rwd.cablock := ifelse(#Pcat1==1, 1, 0)
rwd.cblock := ifelse(#Pd1==1 || #Pe1==1 || #Pdf1==1 || #Pef1==1, 1, 0)
rwd.dblock := ifelse(#Pfdf1==1 || #Pfdt1==1 || #Pfef11==1, 1, 0)
```

B.6 Source code of figure 4.3

```
place Pa
place Pb

place Pcat1
place Pcat2
place Pcat3

place Pcaf1
place Pcaf2
place Pcaf3

place Pcbt1
place Pcbt2
place Pcbt3
place Pcbf1
place Pcbf2
place Pcbf3

place Pd1
place Pd2

place Pe1
place Pe2

place Pfdt1
place Pfdt2
place Pfdt3
place Pfdf1
place Pfdf2
place Pfdf3

place Pfet1
place Pfet2
place Pfet3
```

```
place Pfef1
place Pfef2
place Pfef3


exp a (guard = ga)
exp b (guard = gb)

exp ca (guard = gca1)
exp cafail(guard = gca1)
imm cat1 (guard = gca2)
imm cat2 (guard = gca3)
imm caf1 (guard = gca2)
imm caf2 (guard = gca3)

exp cb (guard = gcb1)
exp cbfail (guard = gcb1)
imm cbt1 (guard = gcb2)
imm cbt2 (guard = gcb3)
imm cbf1 (guard = gcb2)
imm cbf2 (guard = gcb3)

exp c (guard = gc)
exp cf1 (guard = gc)
exp cf2 (guard = gc)
exp cf3 (guard = gc)
exp cf4 (guard = gc)

imm d (guard = gd)
imm e (guard = ge)

exp dt (guard = gd1)
exp df (guard = gd1)
imm fdt1 (guard = gd2)
imm fdt2 (guard = gd3)
imm fdf1 (guard = gd2)
imm fdf2 (guard = gd3)

exp et (guard = ge1)
exp ef (guard = ge1)
imm fet1 (guard = ge2)
imm fet2 (guard = ge3)
imm fef1 (guard = ge2)
imm fef2 (guard = ge3)

exp f
exp ff
exp f1
exp f2
exp f3


arc a to Pa
arc b to Pb

arc Pa to ca
arc ca to Pcat1
arc Pcat1 to cat1
arc cat1 to Pcat2
arc Pcat2 to cat2
arc cat2 to Pcat3
```

```
arc Pa to cafail
arc cafail to Pcaf1
arc Pcaf1 to caf1
arc caf1 to Pcaf2
arc Pcaf2 to caf2
arc caf2 to Pcaf3

arc Pb to cb
arc cb to Pcbt1
arc Pcbt1 to cbt1
arc cbt1 to Pcbt2
arc Pcbt2 to cbt2
arc cbt2 to Pcbt3

arc Pb to cbfail
arc cbfail to Pcbf1
arc Pcbf1 to cbf1
arc cbf1 to Pcbf2
arc Pcbf2 to cbf2
arc cbf2 to Pcbf3

arc Pcat3 to c
arc Pcbt3 to c
arc Pcat3 to cf1
arc Pcbt3 to cf1
arc Pcaf3 to cf2
arc Pcbt3 to cf2
arc Pcat3 to cf3
arc Pcbf3 to cf3
arc Pcaf3 to cf4
arc Pcbf3 to cf4

arc c to Pd1
arc c to Pe1
arc Pd1 to d
arc d to Pd2
arc Pe1 to e
arc e to Pe2

arc Pd2 to dt
arc Pd2 to df

arc df to Pfdf1
arc Pfdf1 to fdf1
arc fdf1 to Pfdf2
arc Pfdf2 to fdf2
arc fdf2 to Pfdf3

arc dt to Pfdt1
arc Pfdt1 to fdt1
arc fdt1 to Pfdt2
arc Pfdt2 to fdt2
arc fdt2 to Pfdt3

arc Pe2 to et
arc Pe2 to ef

arc ef to Pfef1
arc Pfef1 to fef1
arc fef1 to Pfef2
arc Pfef2 to fef2
arc fef2 to Pfef3
```

```
arc et to Pfet1
arc Pfet1 to fet1
arc fet1 to Pfet2
arc Pfet2 to fet2
arc fet2 to Pfet3

arc Pfdf3 to f1
arc Pfef3 to f1
arc Pfdt3 to f2
arc Pfef3 to f2
arc Pfdf3 to f3
arc Pfet3 to f3

arc Pfdt3 to f
arc Pfet3 to f
arc Pfdt3 to ff
arc Pfet3 to ff


ga := #Pa < 2
gb := #Pb < 2

gca1 := #Pcat1 == 0 && #Pcaf1 == 0
gca2 := #Pcat2 == 0 && #Pcaf2 == 0
gca3 := #Pcat3 == 0 && #Pcaf3 == 0

gcb1 := #Pcbt1 == 0 && #Pcbf1 == 0
gcb2 := #Pcbt2 == 0 && #Pcbf2 == 0
gcb3 := #Pcbt3 == 0 && #Pcbf3 == 0

gc := #Pd1==0 && #Pe1==0
gd := #Pd2==0
ge := #Pe2==0

gd1 := #Pfdf1==0 && #Pfdt1==0
gd2 := #Pfdf2==0 && #Pfdt2==0
gd3 := #Pfdf3==0 && #Pfdt3==0

ge1 := #Pfef1==0 && #Pfet1==0
ge2 := #Pfef2==0 && #Pfet2==0
ge3 := #Pfef3==0 && #Pfet3==0

rwd.f := ifelse(#Pfdt3==1 && #Pfet3==1, f.rate, 0)
rwd.ablock := ifelse(#Pa == 2, a.rate, 0)
rwd.cablock := ifelse(#Pcat1==1 || #Pcaf1==1, 1, 0)
rwd.cblock := ifelse(#Pd1==1 || #Pe1==1, 1, 0)
rwd.dblock := ifelse(#Pfdf1==1 || #Pfdt1==1, 1, 0)
```

B.7 Source code of figure 4.4

```
place Pa
place Pb

place Pcat1
place Pcat2
place Pcat3

place Pcaf1
place Pcaf2
place Pcaf3

place Pcbt1
place Pcbt2
place Pcbt3

place Pcbf1
place Pcbf2
place Pcbf3

place Pd1
place Pd2

place Pe1
place Pe2

place Pdf1
place Pdf12

place Pef1
place Pef12

place Pfdt1
place Pfdt2
place Pfdt3
place Pfdf1
place Pfdf2
place Pfdf3

place Pfet1
place Pfet2
place Pfet3
```

```
place Pfef1
place Pfef2
place Pfef3

place Pfdf11
place Pfdf12
place Pfdf13
place Pfef11
place Pfef12
place Pfef13


exp a (guard = ga)
exp b (guard = gb)

exp ca (guard = gca1)
exp cafail(guard = gca1)
imm cat1 (guard = gca2)
imm cat2 (guard = gca3)
imm caf1 (guard = gca2)
imm caf2 (guard = gca3)

exp cb (guard = gcb1)
exp cbfail (guard = gcb1)
imm cbt1 (guard = gcb2)
imm cbt2 (guard = gcb3)
imm cbf1 (guard = gcb2)
imm cbf2 (guard = gcb3)

exp c (guard = gc)
exp cf1 (guard = gc)
exp cf2 (guard = gc)
exp cf3 (guard = gc)
exp cf4 (guard = gc)

imm d (guard = gd)
imm d1 (guard = gd)

imm e (guard = ge)
imm e1 (guard = ge)

exp dt (guard = gd1)
exp df (guard = gd1)
imm fdt1 (guard = gd2, priority = 1)
imm fdt2 (guard = gd3, priority = 1)
imm fdf1 (guard = gd2, priority = 1)
imm fdf2 (guard = gd3, priority = 1)

exp df1 (guard = gd1)
imm fdf11 (guard = gd2)
imm fdf12 (guard = gd3)

exp et (guard = ge1)
exp ef (guard = ge1)
imm fet1 (guard = ge2, priority = 1)
imm fet2 (guard = ge3, priority = 1)
imm fef1 (guard = ge2, priority = 1)
imm fef2 (guard = ge3, priority = 1)
```

```
exp ef1 (guard = ge1)
imm fef11 (guard = ge2)
imm fef12 (guard = ge3)

imm mf1
imm mf12
imm mf13

imm mf2
imm mf22
imm mf23

imm mff1
imm mff2

exp f
exp ff
imm f1
imm f2
imm f3
imm ff1


arc a to Pa
arc b to Pb

arc Pa to ca
arc ca to Pcat1
arc Pcat1 to cat1
arc cat1 to Pcat2
arc Pcat2 to cat2
arc cat2 to Pcat3

arc Pa to cafail
arc cafail to Pcaf1
arc Pcaf1 to caf1
arc caf1 to Pcaf2
arc Pcaf2 to caf2
arc caf2 to Pcaf3

arc Pb to cb
arc cb to Pcbt1
arc Pcbt1 to cbt1
arc cbt1 to Pcbt2
arc Pcbt2 to cbt2
arc cbt2 to Pcbt3

arc Pb to cbfail
arc cbfail to Pcbf1
arc Pcbf1 to cbf1
arc cbf1 to Pcbf2
arc Pcbf2 to cbf2
arc cbf2 to Pcbf3


arc Pcat3 to c
arc Pcbt3 to c
arc Pcat3 to cf1
arc Pcbt3 to cf1
arc Pcaf3 to cf2
arc Pcbt3 to cf2
```

```
arc Pcat3 to cf3
arc Pcbf3 to cf3
arc Pcaf3 to cf4
arc Pcbf3 to cf4

arc c to Pd1
arc c to Pe1
arc Pd1 to d
arc d to Pd2
arc Pe1 to e
arc e to Pe2

arc cf1 to Pdf1
arc cf1 to Pef1
arc Pdf1 to d1
arc d1 to Pdf12
arc Pef1 to e1
arc e1 to Pef12

arc cf2 to Pdf1
arc cf2 to Pef1

arc cf3 to Pdf1
arc cf3 to Pef1

arc cf4 to Pdf1
arc cf4 to Pef1

arc Pd2 to dt
arc Pd2 to df

arc df to Pfdf1
arc Pfdf1 to fdf1
arc fdf1 to Pfdf2
arc Pfdf2 to fdf2
arc fdf2 to Pfdf3

arc dt to Pfdt1
arc Pfdt1 to fdt1
arc fdt1 to Pfdt2
arc Pfdt2 to fdt2
arc fdt2 to Pfdt3


arc Pe2 to et
arc Pe2 to ef

arc ef to Pfef1
arc Pfef1 to fef1
arc fef1 to Pfef2
arc Pfef2 to fef2
arc fef2 to Pfef3

arc et to Pfet1
arc Pfet1 to fet1
arc fet1 to Pfet2
arc Pfet2 to fet2
arc fet2 to Pfet3
```

```
arc Pfdt1 to mf1
arc Pfef1 to mf1
arc Pfdf1 to mf12
arc Pfet1 to mf12
arc Pfdf1 to mf13
arc Pfef1 to mf13

arc Pfdt2 to mf2
arc Pfef2 to mf2
arc Pfdf2 to mf22
arc Pfet2 to mf22
arc Pfdf2 to mf23
arc Pfef2 to mf23


arc Pdf12 to df1
arc df1 to Pfdf11
arc Pfdf11 to fdf11
arc fdf11 to Pfdf12
arc Pfdf12 to fdf12
arc fdf12 to Pfdf13

arc Pef12 to ef1
arc ef1 to Pfef11
arc Pfef11 to fef11
arc fef11 to Pfef12
arc Pfef12 to fef12
arc fef12 to Pfef13

arc Pfdf13 to ff1
arc Pfef13 to ff1

arc Pfdf11 to mff1
arc Pfef11 to mff1
arc Pfdf12 to mff2
arc Pfef12 to mff2

arc Pfdf3 to f1
arc Pfef3 to f1
arc Pfdt3 to f2
arc Pfef3 to f2
arc Pfdf3 to f3
arc Pfet3 to f3

arc Pfdt3 to f
arc Pfet3 to f
arc Pfdt3 to ff
arc Pfet3 to ff


ga := #Pa < 2
gb := #Pb < 2

gca1 := #Pcat1 == 0 && #Pcaf1 == 0
gca2 := #Pcat2 == 0 && #Pcaf2 == 0
gca3 := #Pcat3 == 0 && #Pcaf3 == 0
```

```
gcb1 := #Pcbt1 == 0 && #Pcbf1 == 0
gcb2 := #Pcbt2 == 0 && #Pcbf2 == 0
gcb3 := #Pcbt3 == 0 && #Pcbf3 == 0

gc := #Pd1==0 && #Pe1==0 && #Pdf1==0 && #Pef1==0
gd := #Pd2==0 && #Pdf12==0
ge := #Pe2==0 && #Pef12==0

gd1 := #Pfdf1==0 && #Pfdt1==0 && #Pfdf11==0
gd2 := #Pfdf2==0 && #Pfdt2==0 && #Pfdf12==0
gd3 := #Pfdf3==0 && #Pfdt3==0 && #Pfdf13==0

ge1 := #Pfef1==0 && #Pfet1==0 && #Pfef11==0
ge2 := #Pfef2==0 && #Pfet2==0 && #Pfef12==0
ge3 := #Pfef3==0 && #Pfet3==0 && #Pfef13==0

rwd.f := ifelse(#Pfdt3==1 && #Pfet3==1, f.rate, 0)
rwd.ablock := ifelse(#Pa==2, 1, 0)
rwd.cablock := ifelse(#Pcat1==1 || #Pcaf1==1, 1, 0)
rwd.cblock := ifelse(#Pd1==1 || #Pe1==1 || #Pdf1==1 || #Pef1==1, 1, 0)
rwd.dblock := ifelse(#Pfdt1==1 || #Pfdf1==1 || #Pfef11==1, 1, 0)
```

Bibliography

[1] Hui-Yu Zhang, Qing-Xin Chen, and Ning Mao. 2016. “System performance analysis of flexible flow shop with match processing constraint.” International Journal of Production Research, 2016, Vol. 54, No. 20, pp. 6052-6070.

[2] 岡村寛之, 土肥正 (2016) マルコフ再生確率ペトリネットによる信頼性評価事例，日本信頼性学会誌（学会誌特集：情報通信システムにおける信頼性モデル研究の動向）, vol. 38, no. 6, pp. 340-349.