# Petri net models of class of asynchronous communication mechanisms

Fei Xia and Ian Clark, Department of Electronic and Electrical Engineering, King's College London.

## 1. Abstract

This report describes Petri net models developed for computer-supported analysis of the behaviour of a class of communication mechanisms including the Simpson four-slot technique for transferring data between fully asynchronous processes. These models facilatate an automated state-based alternative analysis approach to time domain simulation, testing and manual reasoning, previously used to verify these kinds of mechanisms. They also make it possible to extend the models to offer more complex mechanisms and to include the effects of metastability. The results of detailed analyses are included in this report.

## 2. Introduction

The use of fully asynchronous processes is advantageous in many hard real-time distributed computer systems. For instance, the complete elimination of time interference in data communication between concurrent processes makes it possible to accurately predict the temporal progress of each process in the system because the timing of each one is completely independent. The class of data communication mechanisms described and analysed by Simpson [1, 2], provides several general communications methods for transferring data between such completely asynchronous processes. The mechanisms link a writing process to a reading process in such a way that the writer can always write a data item without having to wait for the reader, and the reader can always read a data item without having to wait for the writer. They incorporate from one to four data areas (slots), implied to be in memory shared between the writer and the reader. Some of the mechanisms have logic to steer the writer and reader to prevent them from accessing the same slot simultaneously. The reader and writer are not subject to any relative timing constraints and thus are completely temporally independent without the need for synchronisation, wait states or arbitration.

Of these mechanisms, it has been shown that only the four-slot is able to preserve data coherence under all operating conditions [1, 2] except for device abnormalities such as metastable states of control variables. The method previously used to analyse these mechanisms is a specially developed role-transition approach which is a novel notation to assist a process of exhaustive reasoning [2]. This was supplemented by extensive testing of prototype devices [3]. The proof that the four-slot mechanism does ensure data coherence and freshness given in [2] involved a laborious manual search through the whole state space, requiring great care to avoid erroneous or incomplete conclusions. This method approached its limit of use in [2] and [3] in so far that any extension of system complexity or investigation of additional properties (such as the system behaviour if some control variables become metastable) would require so much involvement of human reasoning as to be impractical. In other words, such manual techniques are not easily extendible or repeatable. Therefore, an alternative, extendible, technique based on Petri net models has been developed and is described in this paper.

The theory of Petri nets, first introduced in the early 1960s, has evolved into a comprehensive form [4]. The extensive literature of Petri nets includes many research results on their use in the area of concurrent systems, typified by distributed control systems, manufacturing, etc. They have been used in the modelling and analysis of many aspects of systems [5, 6, 7]. Petri net theory and techniques are also continuously being developed [8, 9], providing a more versatile and practical analysis tool.

This report presents Petri net models of asynchronous data communication mechanisms including the Simpson four-slot mechanism and describes their analysis, using the reachability search technique introduced in [4] as the main tool. The analysis includes verification of the results given in [2] and additional investigations of data freshness and of the effects of metastability on the four-slot mechanism. Petri net notation used throughout the paper follows the convention adapted in [4], of which a brief review is given below.

The elements of Petri nets are places, transitions, markings (tokens) and arcs. Formally, a basic Petri net structure is a 4-tuple $(P, T, I, O)$. $P = \{p_1, p_2, ..., p_n\}$ is a finite set of places, $n \geq 0$. $T = \{t_1, t_2, ..., t_l\}$ is a finite set of transitions, $l \geq 0$. $P \cap T = \varnothing$ (i.e. $P$ and $T$ are disjoint). $I$: $T \longrightarrow P$ is the input function. $O$: $T \longrightarrow P$ is the output function. $I$ and $O$ together constitute the arcs. A marking $M$ of a Petri net ($P$, $T$, $I$, $O$) is a function from the set of places $P$ to the nonnegative integers $\mathbf{N}$. $M$: $P \longrightarrow \mathbf{N}$, where $M(p_i)$ is the number of tokens in place $p_i$, $i = 1, ..., n$. $M$ can alternatively be defined as a vector of $n$ positive integers, $M = (m_1, m_2, ..., m_n)$ where $m_i$ is the number of tokens in place $p_i$, $i = 1, ..., n$. These two definitions are obviously equivalent given $M(p_i) = m_i$. A marked Petri net is a Petri net structure and a marking ($P$, $T$, $I$, $O$, $M$).

In a Petri net graph, structural information is given by places (usually drawn as circles), transitions (usually drawn as rectangles or bars) and arcs (arrowed lines or curves). These together constitute a complete graph for any particular Petri net in its static sense. Oriented arcs connect transitions with places according to the input and output function sets. An arc points from a transition $t_i$ to a place $p_j$ if $O(t_i) = \{..., p_j, ...\}$ or $p_j$ is one of the output places of $t_i$. An arc points from a place $p_j$ to a transition $t_i$ if $I(t_i) = \{..., p_j, ...\}$ or $p_j$ is one of the input places of $t_i$. A marking is represented by a set of tokens (solid disks) deposited in zero or more of the places of a Petri net. It specifies the current state of the net. The state of the net varies with firings of transitions. A transition may fire if and only if every one of its input places has at least one token. Such a transition is also referred to as being enabled. The firing of an enabled transition results in the reduction by one of the number of tokens at every one of its input places and the addition by one of the number of tokens at every one of its output places.

This paper is intended to be read in conjunction with [1] and [2] which provide more detailed background information on the mechanisms including their practical and theoretical significance.

# 3. The mechanisms, their models, and analysis results

## 3.1 Modelling considerations

Since we are interested in data coherence and freshness issues any model must reflect these characteristics of the mechanisms clearly and unambiguously. Data coherence as defined in [1] can be paraphrased as follows: “The reader is obtaining correct data so long as the writing and reading operations do not overlap in time on a single data slot. If such an overlap occurs data coherence is lost.” Data freshness as defined in [1] is more complicated when access procedures with multiple steps are concerned. The reader is considered to be obtaining fresh data so long as it is reading from a slot containing data which was either the freshest or the previous freshest at the beginning of the reading sequence, or else new freshest data, arriving after the beginning of the reading sequence. This excludes re-reading of data older than the last read.

Both of these properties require careful modelling of each step of the reading and writing actions including unambiguous representations of the beginning and end of both the direct writing and reading access to the slots. Other than this the models of both sides of the mechanisms should be kept to the simplest form.

The basic states of these mechanisms can be divided into several groups: The accessing/not accessing of one of the slots by the writing/reading sides; the binary value of a control variable; and the beginning and end of each step of the writing and reading access procedures. All these will be represented with binary places (places whose markings are either zero or one) and the changes of these individual values represented with transitions. This way the current state of the mechanisms is always clear and any property which can be represented with a set of markings can be analysed. These include data coherence and freshness.

### 3.1.1 Writing and reading data

The actions of data reading and writing are represented in the way shown in Figure 1, Figure 3, Figure 6, etc. For each data slot, there is one place representing writing, one representing reading, one representing not writing, and one representing not reading. There is one trasition representing starting writing, one representing finishing writing, one representing starting reading, and one representing finishing reading. Therefore there are two cycles of action for each slot. The transitions may have additional input or output places representing related states of control variables.

### 3.1.2 Global control variables

Some of the control variables in the two- and four-slot mechanisms are shared between the writer and the reader. These are sometimes set to certain values by one of the sides and referenced in value by another. Each control variable is represented by two binary places, one signifying the value of 1 and the other the value of 0. A transition represents the change of value from 0 to 1 and another represents the change of value from 1 to 0. Therefore such a control variable is represented with two places and two transitions. The modification of control variable values is represented in the way shown in Figure 15, Listing 13 and Listing 14. In the case of the three-value control variables of the three-slot mechanism this is done in a similar fashion (Figure 11). Here a value table is directly translated into a Petri net by having a transition representing each line of the value table. With this representation each control variable has a value of either 1 or 0 throughout all the distinct states of the system.

### 3.1.3 Local control variables

Some of the writing and reading access procedures include local control variables. Since we employ step starting and ending places to isolate the individual steps of these procedures these variables only need to have a correctly represented value in the model while in the appropriate steps, although in the actual system algorithms these variables have valid values throughout the process. One way of reducing the number of places and transitions and hence the size of the models is to have the internal control variables take values only during the steps when these values affect the actions of the mechanism. The way this is done is shown in Figure 7, Figure 8 and Figure 9. Here the value of control variable *index* is set during *w*0, made use of in *wr* without modification, and made use of and cleared in *w*1. Because the next use of *index* will happen after the next *w*0 where a new value will be assigned to it, the temporary uncertainty of the value of *index* does not affect the validity of the model.

### 3.1.4 Step starting and ending places

Both the writing and reading access procedures consists of well defined sequential actions. These distinct steps of action should be isolated within the model of each access procedure so that all timing intersections between the writer and reader can be represented. This is represented by places where a previous step's end deposits a token, thereby enabling the next step to begin. There is no such steps in the model of the one-slot mechanism as there is only one step in each cycle of the writing and the reading access procedure.

## 3.2 The one-slot mechanism

The one-slot mechanism, as defined in [1], is shown in Listing 1. Its Petri net model is shown in Figure 1. The full definition of the Petri net model is shown in Listing 2.

An analysis of the mechanism has been carried out in the form of a reachability search of the Petri net model. The system has four distinct operating states (markings): [1, 0, 1, 0] (not writing, not reading), [0, 1, 1, 0] (writing, not reading), [1, 0, 0, 1] (not writing, reading), and [1, 0, 1, 0] (writing, reading). Data coherence is lost in the last state.

**Listing 1 The one-slot mechanism**

**mechanism** *one slot*;

  **var** *data*: **data** := *null*;

  **procedure** *write* (*item*: **data**);
  **begin** *data* := *item* **end**;

  **procedure** *read*: **data**;
  **begin** *read* := *data* **end**;

**end.**

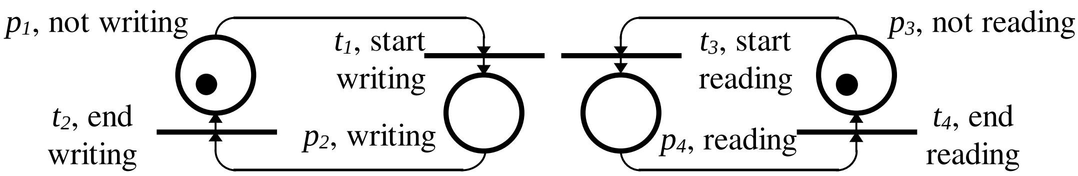


**Figure 1 Petri net model of the one-slot mechanism**

**Listing 2 Petri net model of the one-slot mechanism**

structure:

$P = \{p_i: i = 1\sim4\}$ $\quad$ $T = \{t_j: j = 1\sim4\}$
$I(t_1) = \{p_1\}$ $\quad$ $O(t_1) = \{p_2\}$
$I(t_2) = \{p_2\}$ $\quad$ $O(t_2) = \{p_1\}$
$I(t_3) = \{p_3\}$ $\quad$ $O(t_3) = \{p_4\}$
$I(t_4) = \{p_4\}$ $\quad$ $O(t_4) = \{p_3\}$

initial state:

$M = \{m_i: i = 1\sim4\}$
$m_1 = m_3 = 1$; $\quad$ $m_2 = m_4 = 0$.

modelling:

$m_1 = 1 \rightarrow$ not writing $\quad$ $t_1$: start writing
$m_2 = 1 \rightarrow$ writing $\quad$ $t_2$: end writing
$m_3 = 1 \rightarrow$ not reading $\quad$ $t_3$: start reading
$m_4 = 1 \rightarrow$ reading $\quad$ $t_4$: end reading

## 3.3 The two-slot mechanism

The two-slot mechanism, as defined in [1], is shown in Listing 3. The full definition of its Petri net model is shown in 2 parts in Listing 4 and Listing 5. It is also shown in five parts in Figure 2~Figure 6.

An analysis of the mechanism has been carried out in the form of a reachability search of the Petri net model. 48 distinct operating states were found for the mechanism. Two of these are faulty showing lost data coherence:

1. [0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0] (*latest* = 1, write *index* = 0, read *index* = 0, writing slot 0, reading slot 0). One of the shortest routes (those requiring the smallest number of transition firings) to arrive at this state from the initial state is: $t_1 \rightarrow t_6 \rightarrow t_8 \rightarrow t_{13} \rightarrow t_{17} \rightarrow t_{11} \rightarrow t_3 \rightarrow t_5$ (write *index* = 1 $\rightarrow$ start writing slot 1 $\rightarrow$ end writing slot 1 $\rightarrow$ read *index* = 0 $\rightarrow$ start reading slot 0 $\rightarrow$ *latest* =1 $\rightarrow$ write *index* = 0 $\rightarrow$ start writing slot 0).
2. [0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0] (*latest* = 0, write *index* = 1, read *index* = 1, writing slot 1, reading slot 1) and one of the shortest route to arrive this state from the initial state is: $t_1 \rightarrow t_6 \rightarrow t_8 \rightarrow t_{11} \rightarrow t_3 \rightarrow t_5 \rightarrow t_7 \rightarrow t_{15} \rightarrow t_{18} \rightarrow t_{10} \rightarrow t_1 \rightarrow t_6$ (write *index* = 1 $\rightarrow$ start writing slot 1 $\rightarrow$ end writing slot 1 $\rightarrow$ *latest* =1 $\rightarrow$ write *index* = 0 $\rightarrow$ start writing slot 0 $\rightarrow$ finish writing slot 0 $\rightarrow$ read *index* = 1 $\rightarrow$ start reading slot 1 $\rightarrow$ *latest* = 0 $\rightarrow$ write *index* = 1 $\rightarrow$ start writing slot 1).

**Listing 3 The two-slot mechanism**

```
mechanism two slot;

    var       data: array[bit] of data := (null, null);
              latest: bit := 0;

    procedure write (item: data);
    var       index: bit;
    begin
              index := not latest;                {w0}
              data[index] := item;                {wr}
              latest := index                     {w1}
    end;

    procedure read: data;
    var       index: bit;
    begin
              index := latest;                    {r0}
              read := data[index]                 {rd}
    end;

end.
```

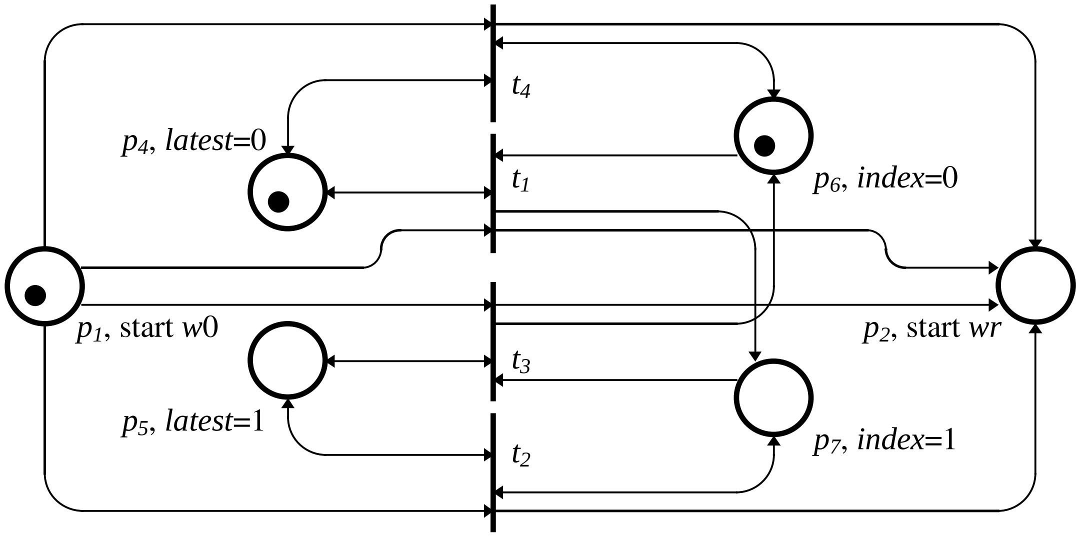


**Figure 2 Petri net model of the two-slot mechanism (Part 1.1** *w*0: *index*:=**not** *latest***)**

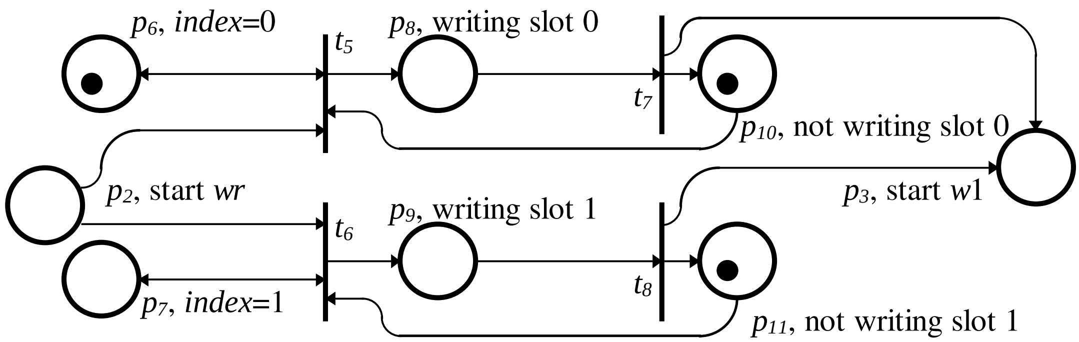


**Figure 3 Petri net model of the two-slot mechanism (Part 1.2** *wr*: *data*[*index*]:=*item***)**

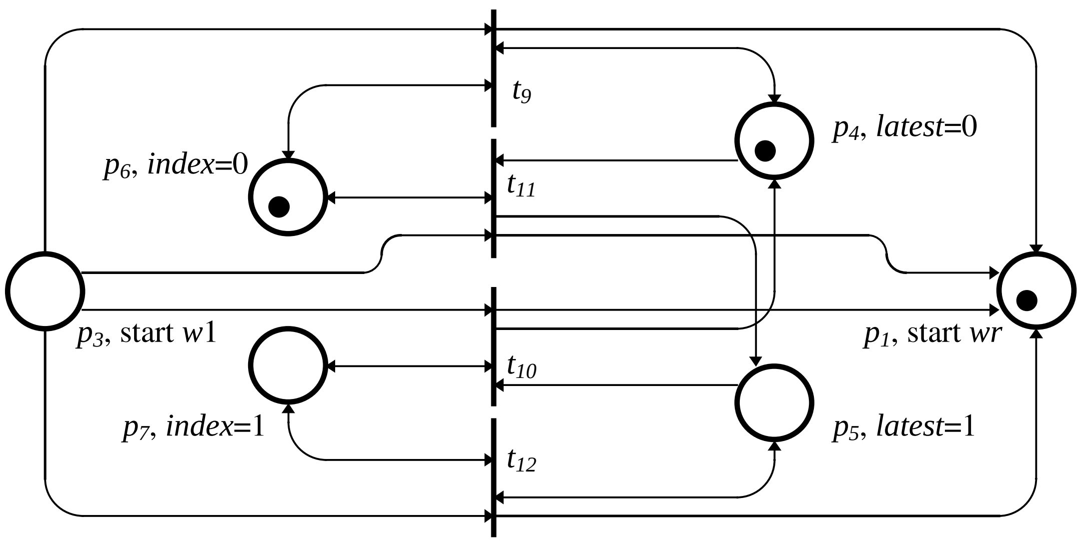


**Figure 4 Petri net model of the two-slot mechanism (Part 1.3** *w*1: *latest*:=*index***)**

**Listing 4 Petri net model of the two-slot mechanism (1)**

(Part 1. Sub-net representing the writer: $\{P_w, T_w, I, O\}$)

structure:

$$\begin{array}{ll} P_w = \{p_i: i = 1\sim11\} & T_w = \{t_j: j = 1\sim12\} \end{array}$$

$$\left.\begin{array}{ll} I(t_1) = \{p_1, p_4, p_6\} & O(t_1) = \{p_2, p_4, p_7\} \\ I(t_2) = \{p_1, p_4, p_7\} & O(t_2) = \{p_2, p_4, p_7\} \\ I(t_3) = \{p_1, p_5, p_7\} & O(t_3) = \{p_2, p_5, p_6\} \\ I(t_4) = \{p_1, p_5, p_6\} & O(t_4) = \{p_2, p_5, p_6\} \end{array}\right\} w0$$

$$\left.\begin{array}{ll} I(t_5) = \{p_2, p_6, p_{10}\} & O(t_5) = \{p_6, p_8\} \\ I(t_6) = \{p_2, p_7, p_{11}\} & O(t_6) = \{p_7, p_9\} \\ I(t_7) = \{p_8\} & O(t_7) = \{p_3, p_{10}\} \\ I(t_8) = \{p_9\} & O(t_8) = \{p_3, p_{11}\} \end{array}\right\} wr$$

$$\left.\begin{array}{ll} I(t_9) = \{p_3, p_4, p_6\} & O(t_9) = \{p_1, p_4, p_6\} \\ I(t_{10}) = \{p_3, p_5, p_6\} & O(t_{10}) = \{p_1, p_4, p_6\} \\ I(t_{11}) = \{p_3, p_4, p_7\} & O(t_{11}) = \{p_1, p_5, p_7\} \\ I(t_{12}) = \{p_3, p_5, p_7\} & O(t_{12}) = \{p_1, p_5, p_7\} \end{array}\right\} w1$$

initial state:

$m_1 = m_4 = m_6 = m_{10} = m_{11} = 1$

$m_2 = m_3 = m_5 = m_7 = m_8 = m_9 = 0$.

model:

$m_1 = 1 \rightarrow$ starting $w0$ $\quad$ $m_2 = 1 \rightarrow$ starting $wr$

$m_3 = 1 \rightarrow$ starting $w1$ $\quad$ $m_4 = 1 \rightarrow$ *latest* = 0

$m_5 = 1 \rightarrow$ *latest* = 1 $\quad$ $m_6 = 1 \rightarrow$ *index* = 0

$m_7 = 1 \rightarrow$ *index* = 1 $\quad$ $m_8 = 1 \rightarrow$ writing slot 0

$m_9 = 1 \rightarrow$ writing slot 1 $\quad$ $m_{10} = 1 \rightarrow$ not writing 0

$m_{11} = 1 \rightarrow$ not writing 1

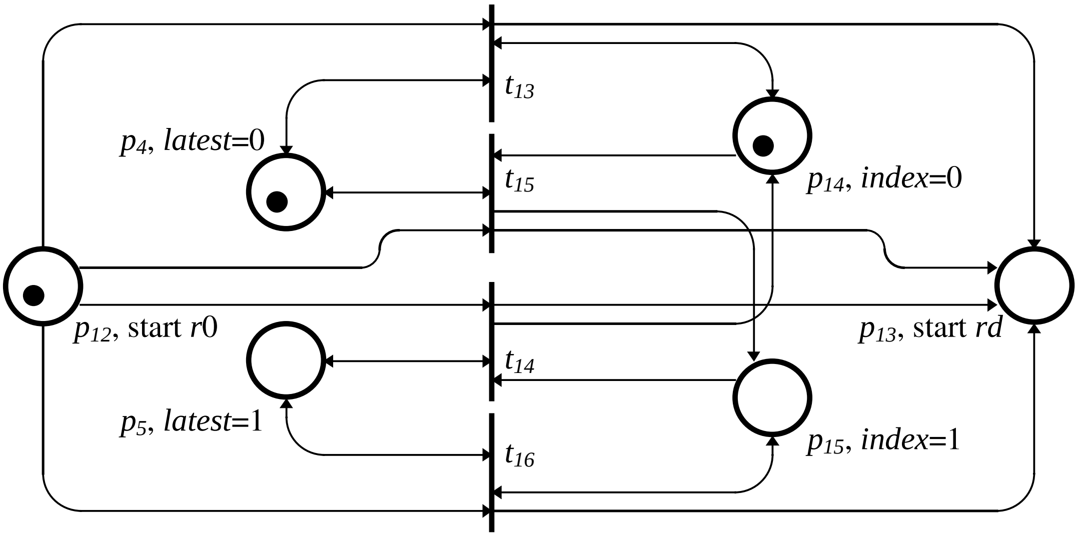


**Figure 5 Petri net model of the two-slot mechanism (Part 2.1** *r*0: *index*:=*latest***)**

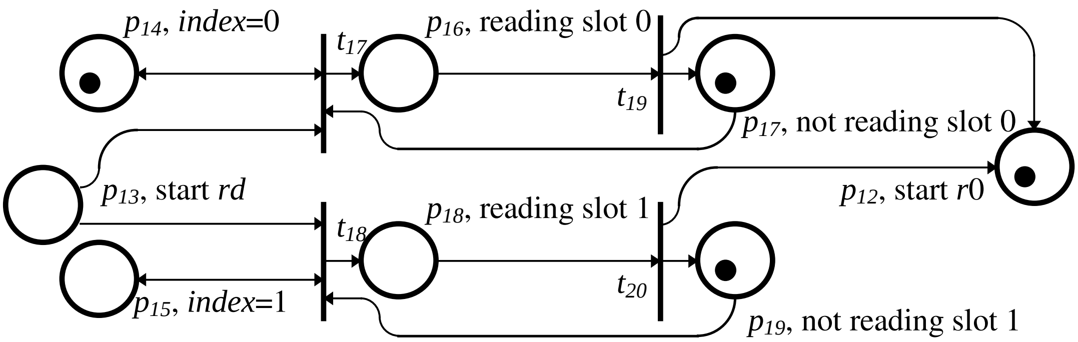


**Figure 6 Petri net model of the two-slot mechanism (Part 2.2** *rd*: *read*:=*data*[*index*]**)**

**Listing 5 Petri net model of the two-slot mechanism (2)**

(Part 2. Sub-net representing the reader: $\{P_r, T_r, I, O\}$)

structure:

$P_r = \{p_i: i = 4, 5, 12\sim19\}$ $T_r = \{t_j: j = 13\sim20\}$

| | | |
|---|---|---|
| $I(t_{13}) = \{p_4, p_{12}, p_{14}\}$ | $O(t_{13}) = \{p_4, p_{13}, p_{14}\}$ | $r0$ |
| $I(t_{14}) = \{p_4, p_{12}, p_{15}\}$ | $O(t_{14}) = \{p_4, p_{13}, p_{14}\}$ | |
| $I(t_{15}) = \{p_5, p_{12}, p_{14}\}$ | $O(t_{15}) = \{p_5, p_{13}, p_{15}\}$ | |
| $I(t_{16}) = \{p_5, p_{12}, p_{15}\}$ | $O(t_{16}) = \{p_5, p_{13}, p_{15}\}$ | |
| $I(t_{17}) = \{p_{13}, p_{14}, p_{17}\}$ | $O(t_{17}) = \{p_{14}, p_{16}\}$ | $rd$ |
| $I(t_{18}) = \{p_{13}, p_{15}, p_{19}\}$ | $O(t_{18}) = \{p_{15}, p_{18}\}$ | |
| $I(t_{19}) = \{p_{16}\}$ | $O(t_{19}) = \{p_{12}, p_{17}\}$ | |
| $I(t_{20}) = \{p_{18}\}$ | $O(t_{20}) = \{p_{12}, p_{19}\}$ | |

initial state: $m_{12} = m_{14} = m_{17} = m_{19} = 1$;
$m_{13} = m_{15} = m_{16} = m_{18} = 0$.

model:

$m_{12} = 1 \rightarrow$ the start of $r0$ $\quad$ $m_{13} = 1 \rightarrow$ the start of $rd$

$m_{14} = 1 \rightarrow$ *index* = 0 $\quad$ $m_{15} = 1 \rightarrow$ *index* = 1

$m_{16} = 1 \rightarrow$ reading slot 0 $m_{17} = 1 \rightarrow$ reading slot 1

$m_{18} = 1 \rightarrow$ not reading 0 $\quad$ $m_{19} = 1 \rightarrow$ not reading 1

### 3.4 The three-slot mechanism

The three-slot mechanism, as defined in [1], is shown in Listing 6. The full definition of its Petri net model is shown in two parts in Listing 7 and Listing 8 and in six parts in Figure 7~Figure 12.

An analysis of the mechanism has been carried out in the form of a reachability search of the Petri net model. 324 distinct operating states were found for the mechanism. Six of these are faulty showing lost data coherence. They are:

Data coherence lost on slot 1:

1. [0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0] (Both $p_{13}$ and $p_{22}$ are marked, indicating reading and writing slot 1 at the same time). One of the shortest routes to arrive at this state from the initial state is:
   $t_4 \rightarrow t_{11} \rightarrow t_{14} \rightarrow t_{19} \rightarrow t_8 \rightarrow t_{12} \rightarrow t_{15} \rightarrow t_{23} \rightarrow t_{27} \rightarrow t_{34} \rightarrow t_3 \rightarrow t_{10} \rightarrow t_{13}. \rightarrow t_{18} \rightarrow t_5 \rightarrow t_{11} \rightarrow t_{14} \rightarrow t_{39} \rightarrow t_{42} \rightarrow t_{25} \rightarrow t_{19} \rightarrow t_1 \rightarrow t_{10} \rightarrow t_{30} \rightarrow t_{37}$
2. [0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0] (Both $p_{13}$ and $p_{22}$ are marked, indicating reading and writing slot 1 at the same time). One of the shortest routes to arrive at this state from the initial state is:
   $t_4 \rightarrow t_{11} \rightarrow t_{14} \rightarrow t_{19} \rightarrow t_8 \rightarrow t_{12} \rightarrow t_{15} \rightarrow t_{26} \rightarrow t_{23} \rightarrow t_{31} \rightarrow t_2 \rightarrow t_{10} \rightarrow t_{13}. \rightarrow t_{18} \rightarrow t_7 \rightarrow t_{12} \rightarrow t_{15} \rightarrow t_{38} \rightarrow t_{41} \rightarrow t_{25} \rightarrow t_{22} \rightarrow t_2 \rightarrow t_{10} \rightarrow t_{29} \rightarrow t_{37}$.

Data coherence lost on slot 2:

1. [0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0] (Both $p_{14}$ and $p_{23}$ are marked, indicating reading and writing slot 2 at the same time). One of the shortest routes to arrive at this state from the initial state is: $t_4 \rightarrow t_{11} \rightarrow t_{14} \rightarrow t_{19} \rightarrow t_8 \rightarrow t_{12} \rightarrow t_{15} \rightarrow t_{26} \rightarrow t_{23} \rightarrow t_6 \rightarrow t_{11} \rightarrow t_{31} \rightarrow t_{38}$.
2. [0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0] (Both $p_{14}$ and $p_{23}$ are marked, indicating reading and writing slot 2 at the same time). One of the shortest routes to arrive at this state from the initial state is: $t_4 \rightarrow t_{11} \rightarrow t_{14} \rightarrow t_{19} \rightarrow t_8 \rightarrow t_{12} \rightarrow t_{15} \rightarrow t_{23} \rightarrow t_6 \rightarrow t_{11} \rightarrow t_{14} \rightarrow t_{27} \rightarrow t_{21}. \rightarrow t_{34} \rightarrow t_1 \rightarrow t_{10} \rightarrow t_{13} \rightarrow t_{39} \rightarrow t_{42} \rightarrow t_{26} \rightarrow t_{17} \rightarrow t_5 \rightarrow t_{11} \rightarrow t_{33} \rightarrow t_{38}$.

Data coherence lost on slot 3:

1. [0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0] (Both $p_{15}$ and $p_{24}$ are marked, indicating reading and writing slot 3 at the same time). One of the shortest routes to arrive at this state from the initial state is:
   $t_4 \rightarrow t_{11} \rightarrow t_{14} \rightarrow t_{19} \rightarrow t_8 \rightarrow t_{12} \rightarrow t_{15} \rightarrow t_{23} \rightarrow t_6 \rightarrow t_{11} \rightarrow t_{14} \rightarrow t_{27} \rightarrow t_{21}. \rightarrow t_8 \rightarrow t_{12} \rightarrow t_{34} \rightarrow t_{39}$
2. [0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0] (Both $p_{15}$ and $p_{24}$ are marked, indicating reading and writing slot 3 at the same time). One of the shortest routes to arrive at this state from the initial state is:
   $t_4 \rightarrow t_{11} \rightarrow t_{14} \rightarrow t_{19} \rightarrow t_8 \rightarrow t_{12} \rightarrow t_{15} \rightarrow t_{26} \rightarrow t_{23} \rightarrow t_{31} \rightarrow t_2 \rightarrow t_{10} \rightarrow t_{13}. \rightarrow t_{38} \rightarrow t_{41} \rightarrow t_{27} \rightarrow t_{18} \rightarrow t_7 \rightarrow t_{12} \rightarrow t_{35} \rightarrow t_{39}$

**Listing 6 The three-slot mechanism**

```
mechanism three slot;

        var         data: array[1..3] of data := (null, null, null);
                    latest, reading: 1..3 := 1, 1;

        procedure write (item: data);
        const differ = ((2, 3, 2), (3, 3, 1), (2, 1, 1));
        var         index: 1..3;
        begin
                    index := differ[latest, reading];          {w0}
                    data[index] := item;                       {wr}
                    latest := index                            {w1}
        end;

        procedure read: data;
        var         index: 1..3;
        begin
                    index := latest;                           {r0}
                    reading := index;                          {r1}
                    read := data[index]                        {rd}
        end;

end.
```

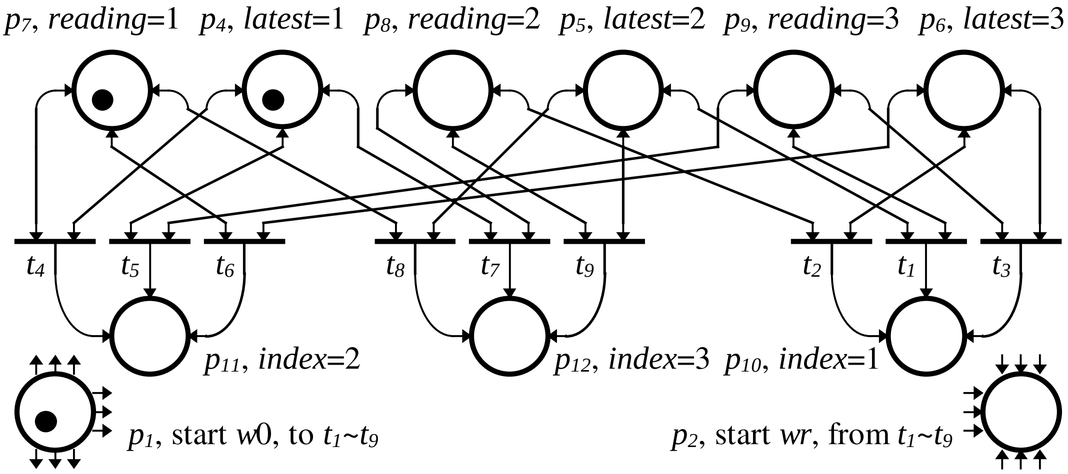


**Figure 7 Petri net model of the three-slot mechanism (Part 1.1** *w*0: *index* := *differ*[*latest*, *reading*]**)**

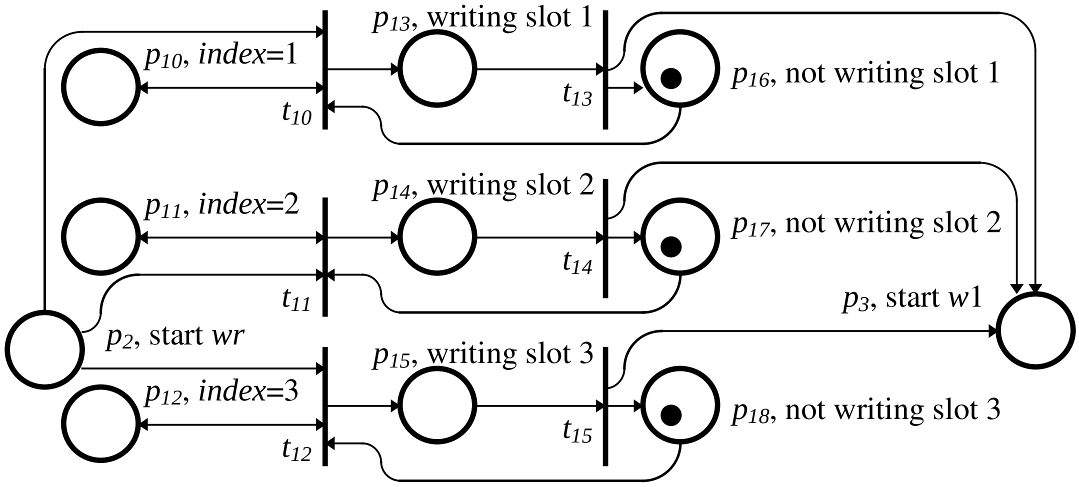


**Figure 8 Petri net model of the three-slot mechanism (Part 1.2** *wr*: *data*[*index*] := *item***)**

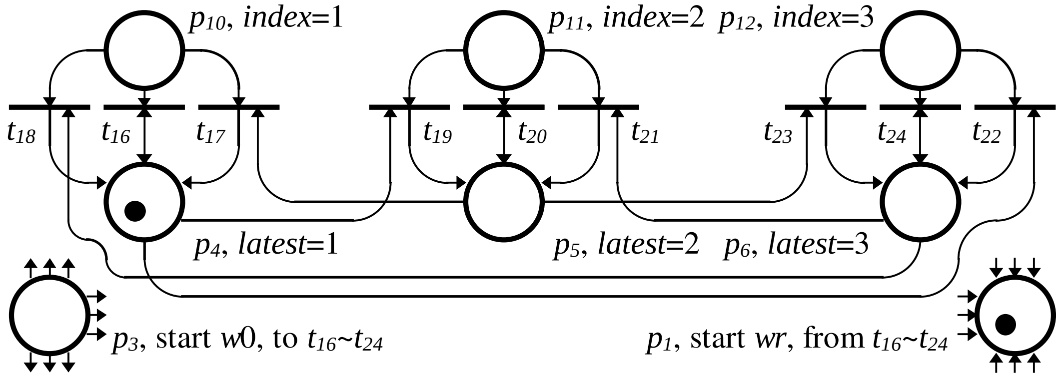


**Figure 9 Petri net model of the three-slot mechanism (Part 1.3** *w*1: *latest* := *index***)**

**Listing 7 Petri net model of the three-slot mechanism (1)**

(Part 1. Sub-net representing the writer: $\{P_w, T_w, I, O\}$)
structure:

$P_w = \{p_i: i = 1\sim18\}$ $T_w = \{t_j: j = 1\sim24\}$

$$\left.\begin{array}{ll}
I(t_1) = \{p_1, p_5, p_9\} & O(t_1) = \{p_2, p_5, p_9, p_{10}\} \\
I(t_2) = \{p_1, p_6, p_8\} & O(t_2) = \{p_2, p_6, p_8, p_{10}\} \\
I(t_3) = \{p_1, p_6, p_9\} & O(t_3) = \{p_2, p_6, p_9, p_{10}\} \\
I(t_4) = \{p_1, p_4, p_7\} & O(t_4) = \{p_2, p_4, p_7, p_{11}\} \\
I(t_5) = \{p_1, p_4, p_9\} & O(t_5) = \{p_2, p_4, p_9, p_{11}\} \\
I(t_6) = \{p_1, p_6, p_7\} & O(t_6) = \{p_2, p_6, p_7, p_{11}\} \\
I(t_7) = \{p_1, p_4, p_8\} & O(t_7) = \{p_2, p_4, p_8, p_{12}\} \\
I(t_8) = \{p_1, p_5, p_7\} & O(t_8) = \{p_2, p_5, p_7, p_{12}\} \\
I(t_9) = \{p_1, p_5, p_8\} & O(t_9) = \{p_2, p_5, p_8, p_{12}\}
\end{array}\right\} w0$$

$$\left.\begin{array}{ll}
I(t_{10}) = \{p_2, p_{10}, p_{16}\} & O(t_{10}) = \{p_{10}, p_{13}\} \\
I(t_{11}) = \{p_2, p_{11}, p_{17}\} & O(t_{11}) = \{p_{11}, p_{14}\} \\
I(t_{12}) = \{p_2, p_{12}, p_{18}\} & O(t_{12}) = \{p_{12}, p_{15}\} \\
I(t_{13}) = \{p_{13}\} & O(t_{13}) = \{p_3, p_{16}\} \\
I(t_{14}) = \{p_{14}\} & O(t_{14}) = \{p_3, p_{17}\} \\
I(t_{15}) = \{p_{15}\} & O(t_{15}) = \{p_3, p_{18}\}
\end{array}\right\} wr$$

$$\left.\begin{array}{ll}
I(t_{16}) = \{p_3, p_4, p_{10}\} & O(t_{16}) = \{p_1, p_4\} \\
I(t_{17}) = \{p_3, p_5, p_{10}\} & O(t_{17}) = \{p_1, p_4\} \\
I(t_{18}) = \{p_3, p_6, p_{10}\} & O(t_{18}) = \{p_1, p_4\} \\
I(t_{19}) = \{p_3, p_4, p_{11}\} & O(t_{19}) = \{p_1, p_5\} \\
I(t_{20}) = \{p_3, p_5, p_{11}\} & O(t_{20}) = \{p_1, p_5\} \\
I(t_{21}) = \{p_3, p_6, p_{11}\} & O(t_{21}) = \{p_1, p_5\} \\
I(t_{22}) = \{p_3, p_4, p_{12}\} & O(t_{22}) = \{p_1, p_6\} \\
I(t_{23}) = \{p_3, p_5, p_{12}\} & O(t_{23}) = \{p_1, p_6\} \\
I(t_{24}) = \{p_3, p_6, p_{12}\} & O(t_{24}) = \{p_1, p_6\}
\end{array}\right\} w1$$

initial state:

$m_1 = m_4 = m_7 = m_{16} = m_{17} = m_{18} = 1$

$m_2 = m_3 = m_5 = m_6 = m_8 = m_9 = m_{10} = m_{11} = m_{12} = m_{13} = m_{14} = m_{15} = 0.$

model:

$m_1 = 1 \rightarrow$ starting $w0$ $m_2 = 1 \rightarrow$ starting $wr$
$m_3 = 1 \rightarrow$ starting $w1$ $m_4 = 1 \rightarrow$ *latest* $= 1$
$m_5 = 1 \rightarrow$ *latest* $= 2$ $m_6 = 1 \rightarrow$ *latest* $= 3$
$m_7 = 1 \rightarrow$ *reading* $= 1$ $m_8 = 1 \rightarrow$ *reading* $= 2$
$m_9 = 1 \rightarrow$ *reading* $= 3$ $m_{10} = 1 \rightarrow$ *index* $= 1$
$m_{11} = 1 \rightarrow$ *index* $= 2$ $m_{12} = 1 \rightarrow$ *index* $= 3$
$m_{13} = 1 \rightarrow$ writing slot 1 $m_{14} = 1 \rightarrow$ writing slot 2
$m_{15} = 1 \rightarrow$ writing slot 3 $m_{16} = 1 \rightarrow$ not writing 1
$m_{17} = 1 \rightarrow$ not writing 2 $m_{18} = 1 \rightarrow$ not writing 3

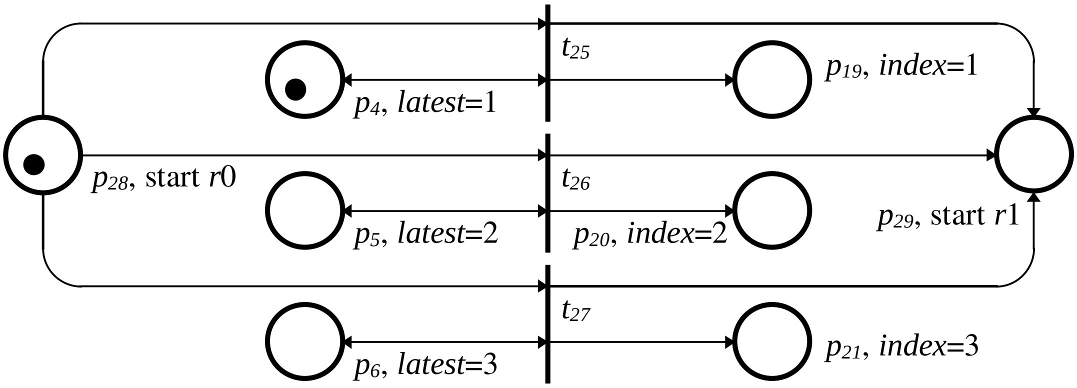


**Figure 10 Petri net model of the three-slot mechanism (Part 2.1** $r0$: *index* := *latest***)**

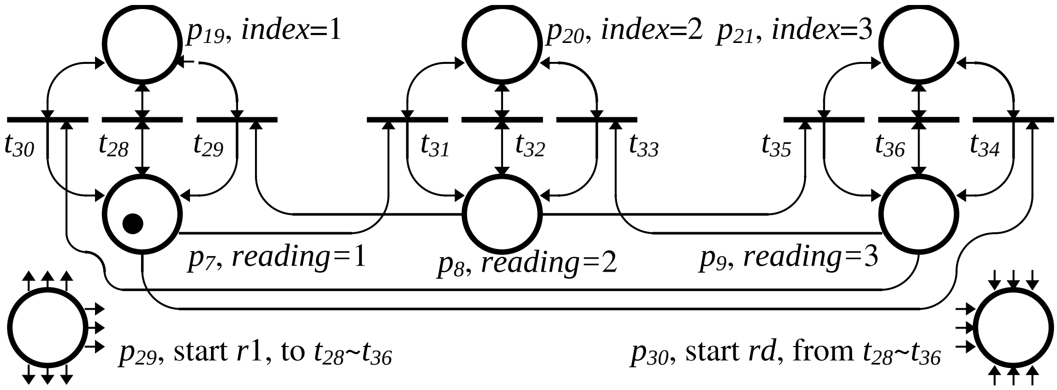


**Figure 11 Petri net model of the three-slot mechanism (Part 2.2** $r1$: *reading* := *index***)**

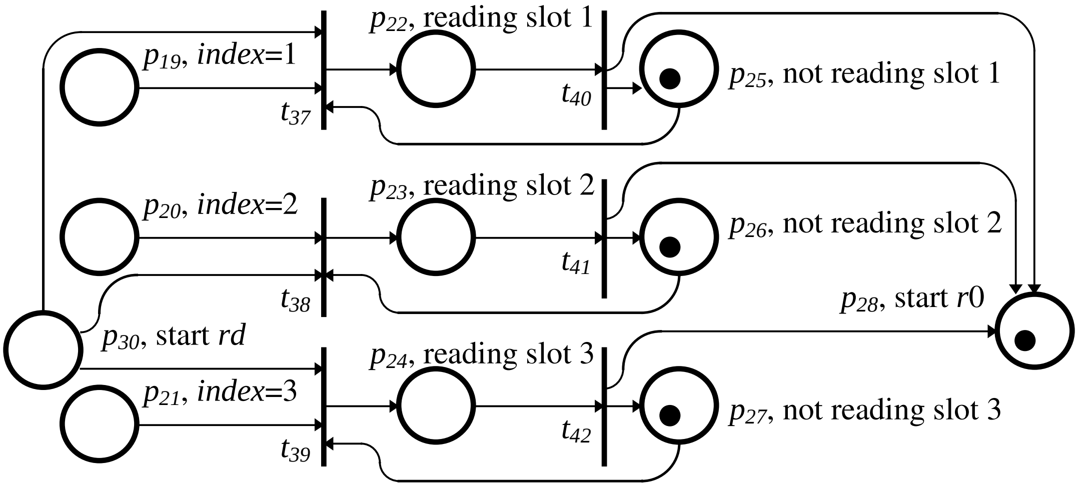


**Figure 12 Petri net model of the three-slot mechanism (Part 2.3** $rd$: *read* := *data*[*index*]**)**

**Listing 8 Petri net model of the three-slot mechanism (2)**

Part 2. Sub-net representing the reader: $\{P_r, T_r, I, O\}$
structure:

$P_r = \{p_i: i = 4\sim6, 7\sim9, 19\sim30\}$ $T_r = \{t_j: j = 25\sim42\}$
$I(t_{25}) = \{p_4, p_{28}\}$ $O(t_{25}) = \{p_4, p_{19}, p_{29}\}$
$I(t_{26}) = \{p_5, p_{28}\}$ $O(t_{26}) = \{p_5, p_{20}, p_{29}\}$ $r0$
$I(t_{27}) = \{p_6, p_{28}\}$ $O(t_{27}) = \{p_6, p_{21}, p_{29}\}$
$I(t_{28}) = \{p_7, p_{19}, p_{29}\}$ $O(t_{28}) = \{p_7, p_{19}, p_{30}\}$
$I(t_{29}) = \{p_8, p_{19}, p_{29}\}$ $O(t_{29}) = \{p_7, p_{19}, p_{30}\}$
$I(t_{30}) = \{p_9, p_{19}, p_{29}\}$ $O(t_{30}) = \{p_7, p_{19}, p_{30}\}$
$I(t_{31}) = \{p_7, p_{20}, p_{29}\}$ $O(t_{31}) = \{p_8, p_{20}, p_{30}\}$
$I(t_{32}) = \{p_8, p_{20}, p_{29}\}$ $O(t_{32}) = \{p_8, p_{20}, p_{30}\}$ $r1$
$I(t_{33}) = \{p_9, p_{20}, p_{29}\}$ $O(t_{33}) = \{p_8, p_{20}, p_{30}\}$
$I(t_{34}) = \{p_7, p_{21}, p_{29}\}$ $O(t_{34}) = \{p_9, p_{21}, p_{30}\}$
$I(t_{35}) = \{p_8, p_{21}, p_{29}\}$ $O(t_{35}) = \{p_9, p_{21}, p_{30}\}$
$I(t_{36}) = \{p_9, p_{21}, p_{29}\}$ $O(t_{36}) = \{p_9, p_{21}, p_{30}\}$
$I(t_{37}) = \{p_{19}, p_{25}, p_{30}\}$ $O(t_{37}) = \{p_{22}\}$
$I(t_{38}) = \{p_{20}, p_{26}, p_{30}\}$ $O(t_{38}) = \{p_{23}\}$
$I(t_{39}) = \{p_{21}, p_{27}, p_{30}\}$ $O(t_{39}) = \{p_{24}\}$ $rd$
$I(t_{40}) = \{p_{22}\}$ $O(t_{40}) = \{p_{25}, p_{28}\}$
$I(t_{41}) = \{p_{23}\}$ $O(t_{41}) = \{p_{26}, p_{28}\}$
$I(t_{42}) = \{p_{24}\}$ $O(t_{42}) = \{p_{27}, p_{28}\}$

initial state: $m_{25} = m_{26} = m_{27} = m_{28} = 1$;
$m_{19} = m_{20} = m_{21} = m_{22} = m_{23} = m_{24} = m_{29} = m_{30} = 0$.

model:
$m_{19} = 1 \rightarrow index = 1$ $m_{20} = 1 \rightarrow index = 2$
$m_{21} = 1 \rightarrow index = 3$ $m_{22} = 1 \rightarrow$ reading slot 1
$m_{23} = 1 \rightarrow$ reading slot 2 $m_{24} = 1 \rightarrow$ reading slot 3
$m_{25} = 1 \rightarrow$ not reading 1 $m_{26} = 1 \rightarrow$ not reading 2
$m_{27} = 1 \rightarrow$ not reading 3 $m_{28} = 1 \rightarrow$ the start of $r0$
$m_{29} = 1 \rightarrow$ the start of $r1$ $m_{30} = 1 \rightarrow$ the start of $rd$

## 3.5 The four-slot mechanism

Several versions of the four-slot mechanism are discussed in [1] and [2]. Here two of them are modelled and analysed in detail. The first is the basic algorithm of the four-slot mechanism and the second is one way of implementing it in hardware.

### 3.5.1 Basic four-slot mechanism algorithm

The basic algorithm of the four-slot mechanism is shown in Listing 9. To simplify matters, the model of each of the steps of the reading and writing access procedures is presented separately.

**Listing 9 Basic algorithm of the four-slot mechanism**

```
mechanism four slot;

    var       data: array[bit, bit] of data := ((null, null), (null, null));
              slot: array[bit] of bit := (0, 0);
              latest, reading: bit := 0, 0;

    procedure write (item: data);
    var       pair, index: bit;
    begin
              pair := not reading;                  {w0}
              index := not slot[pair];              {w1}
              data[pair, index] := item;            {w2}
              slot[pair] := index;                  {w3}
              latest := pair                        {w4}
    end;

    procedure read: data;
    var       pair, index: bit;
    begin
              pair := latest;                       {r0}
              reading := pair;                      {r1}
              index := slot[pair];                  {r2}
              read := data[pair, index]             {r3}
    end;

end.
```

### 3.5.2 Four-slot mechanism, hardware implementation

One way to implement the four-slot mechanism as given in [1] is shown in Listing 10. Here both the writing and reading access procedures are reduced to three steps each. As before models for these steps are presented separately.

**Listing 10 Four-slot mechanism hardware implementation**

| Writing | Reading |
|---|---|
| $wr\colon\ d[n, \overline{s[n]}] := input$ | $r0\colon\ r := l$ |
| $w0\colon\ s[n] := \overline{s[n]}$ | $r1\colon\ v := s$ |
| $w1\colon\ l := n \parallel n := \overline{r}$ | $rd\colon\ output := d[r, v[r]]$ |

The petri net model of this hardware implementation is shown below in six sub nets (Figure 13~Figure 18, Listing 11~Listing 13, and Listing 15~Listing 17). The value table in Listing 14 explains how statements which modify binary variable values are modelled with transitions.

A reachability search has been carried out on the full model. There is no states where any one slot is being written or read at the same time. Essentially this implementation of the four-slot mechanism does maintain data coherence.

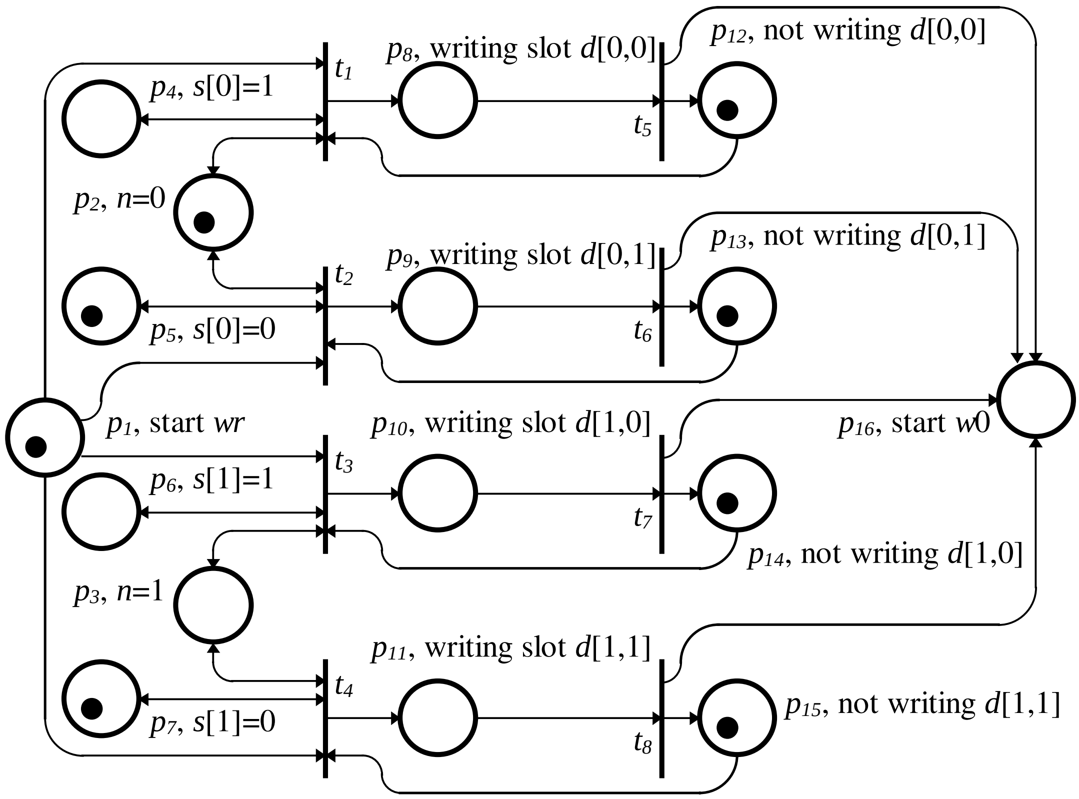


**Figure 13 Petri net model of the four-slot mechanism hardware implementation**
**(Part 1: $d[n, \overline{s[n]}] := input$ )**

**Listing 11 Petri net model of the four-slot mechanism hardware implementation (1)**

(Part 1. Sub-net representing *wr*: $d[n, \overline{s[n]}] := input$ : $\{P_{wr}, T_{wr}, I, O\}$)

structure:

$P_{wr} = \{p_i: i = 1\sim16\}$ $T_{wr} = \{t_j: j = 1\sim8\}$
$I(t_1) = \{p_1, p_2, p_4, p_{12}\}$ $O(t_1) = \{p_2, p_4, p_8\}$
$I(t_2) = \{p_1, p_2, p_5, p_{13}\}$ $O(t_2) = \{p_2, p_5, p_9\}$
$I(t_3) = \{p_1, p_3, p_6, p_{14}\}$ $O(t_3) = \{p_3, p_6, p_{10}\}$
$I(t_4) = \{p_1, p_3, p_7, p_{15}\}$ $O(t_4) = \{p_3, p_7, p_{11}\}$
$I(t_5) = \{p_8\}$ $O(t_5) = \{p_{12}, p_{16}\}$
$I(t_6) = \{p_9\}$ $O(t_6) = \{p_{13}, p_{16}\}$
$I(t_7) = \{p_{10}\}$ $O(t_7) = \{p_{14}, p_{16}\}$
$I(t_8) = \{p_{11}\}$ $O(t_8) = \{p_{15}, p_{16}\}$

initial state:

$m_1 = m_2 = m_5 = m_7 = m_{12} = m_{13} = m_{14} = m_{15} = 1$
$m_3 = m_4 = m_6 = m_8 = m_9 = m_{10} = m_{11} = m_{16} = 0.$

model:

$m_1 = 1 \rightarrow$ starting *wr* $m_2 = 1 \rightarrow n = 0$
$m_3 = 1 \rightarrow n = 1$ $m_4 = 1 \rightarrow s[0] = 1$
$m_5 = 1 \rightarrow s[0] = 0$ $m_6 = 1 \rightarrow s[1] = 1$
$m_7 = 1 \rightarrow s[1] = 0$ $m_8 = 1 \rightarrow$ writing slot $d[0,0]$
$m_9 = 1 \rightarrow$ writing $d[0,1]$ $m_{10} = 1 \rightarrow$ writing $d[1,0]$
$m_{11} = 1 \rightarrow$ writing $d[1,1]$ $m_{12} = 1 \rightarrow$ not writing $d[0,0]$
$m_{13} = 1 \rightarrow$ not writing $d[0,1]$ $m_{14} = 1 \rightarrow$ not writing $d[1,0]$
$m_{15} = 1 \rightarrow$ not writing $d[1,1]$ $m_{16} = 1 \rightarrow$ starting *w*0

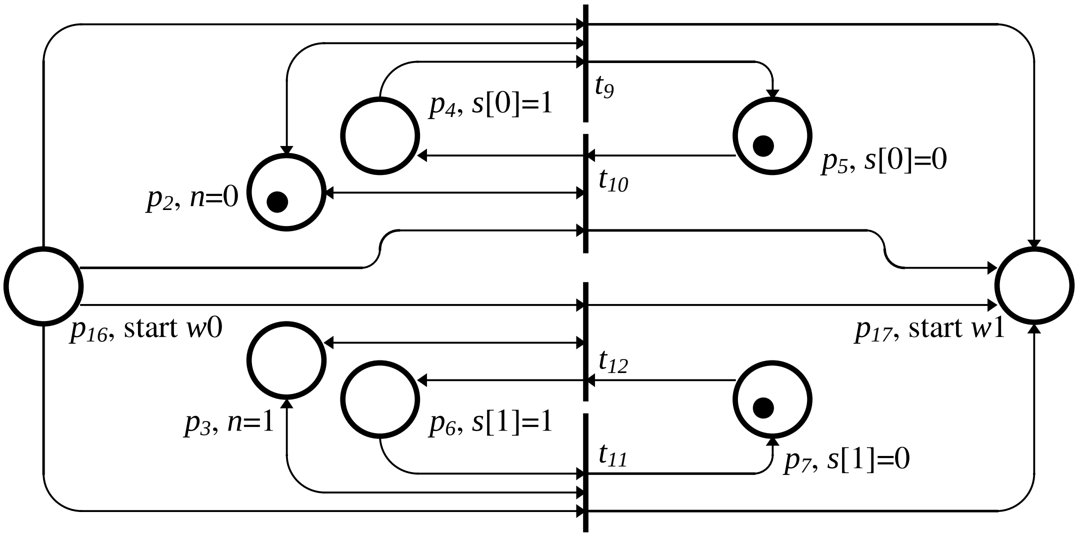


**Figure 14 Petri net model of the four-slot mechanism hardware implementation**
**(Part 2: $s[n] := \overline{s[n]}$ )**

**Listing 12 Petri net model of the four-slot mechanism hardware implementation (2)**

(Part 2. Sub-net representing $w0$: $s[n] := \overline{s[n]}$ : $\{P_{w0}, T_{w0}, I, O\}$)

structure:

| | |
|---|---|
| $P_{w0} = \{p_i: i = 2\text{\textasciitilde}7,16,17\}$ | $T_{w0} = \{t_j: j = 8\text{\textasciitilde}12\}$ |
| $I(t_9) = \{p_2, p_4, p_{16}\}$ | $O(t_9) = \{p_2, p_5, p_{17}\}$ |
| $I(t_{10}) = \{p_2, p_5, p_{16}\}$ | $O(t_{10}) = \{p_2, p_4, p_{17}\}$ |
| $I(t_{11}) = \{p_3, p_6, p_{16}\}$ | $O(t_{11}) = \{p_3, p_7, p_{17}\}$ |
| $I(t_{12}) = \{p_3, p_7, p_{16}\}$ | $O(t_{12}) = \{p_3, p_6, p_{17}\}$ |

initial state: $m_{17} = 0$.

model: $m_{17} = 1 \rightarrow$ the start of $w1$

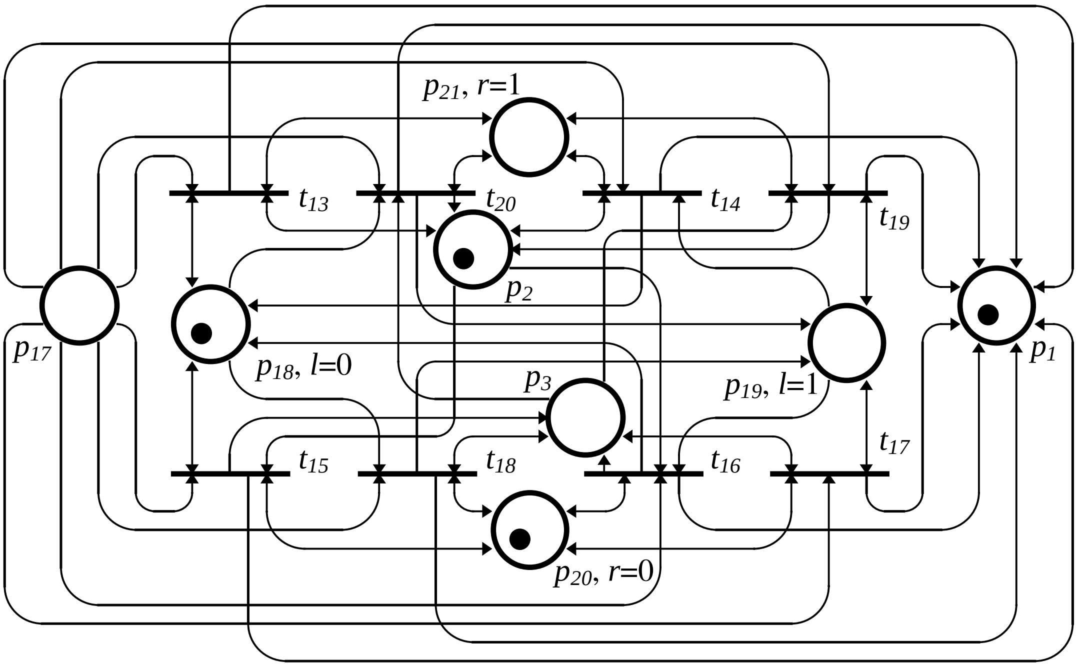


**Figure 15 Petri net model of the four-slot mechanism hardware implementation**
**(Part 3: $l := n \| n := \bar{r}$ )**

**Listing 13 Petri net model of the four-slot mechanism hardware implementation (3)**

(Part 3. Sub-net representing $w1$: $l := n \| n := \bar{r}$ : $\{P_{w1}, T_{w1}, I, O\}$)

structure:

$P_{w1} = \{p_i: i = 1\sim3,17\sim21\}$ $\quad$ $T_{w1} = \{t_j: j = 13\sim20\}$
$I(t_{13}) = \{p_2, p_{17}, p_{18}, p_{21}\}$ $\quad$ $O(t_{13}) = \{p_1, p_2, p_{18}, p_{21}\}$
$I(t_{14}) = \{p_2, p_{17}, p_{19}, p_{21}\}$ $\quad$ $O(t_{14}) = \{p_1, p_2, p_{18}, p_{21}\}$
$I(t_{15}) = \{p_2, p_{17}, p_{18}, p_{20}\}$ $\quad$ $O(t_{15}) = \{p_1, p_3, p_{18}, p_{20}\}$
$I(t_{16}) = \{p_2, p_{17}, p_{19}, p_{20}\}$ $\quad$ $O(t_{16}) = \{p_1, p_3, p_{18}, p_{20}\}$
$I(t_{17}) = \{p_3, p_{17}, p_{19}, p_{20}\}$ $\quad$ $O(t_{17}) = \{p_1, p_3, p_{19}, p_{20}\}$
$I(t_{18}) = \{p_3, p_{17}, p_{18}, p_{20}\}$ $\quad$ $O(t_{18}) = \{p_1, p_3, p_{19}, p_{20}\}$
$I(t_{19}) = \{p_3, p_{17}, p_{19}, p_{21}\}$ $\quad$ $O(t_{19}) = \{p_1, p_2, p_{19}, p_{21}\}$
$I(t_{20}) = \{p_3, p_{17}, p_{18}, p_{21}\}$ $\quad$ $O(t_{20}) = \{p_1, p_2, p_{19}, p_{21}\}$

initial state: $\quad$ $m_{18} = m_{20} = 1$ $\quad$ $m_{19} = m_{21} = 0$.

model: $\quad$ $m_{18} = 1 \rightarrow l = 0$ $\quad$ $m_{19} = 1 \rightarrow l = 1$
$m_{20} = 1 \rightarrow r = 0$ $\quad$ $m_{21} = 1 \rightarrow r = 1$

**Listing 14 Value table of $w1$: $l := n \| n := \bar{r}$ :**

| before statement: | | | after statement: | | | transition |
|---|---|---|---|---|---|---|
| $l$ | $n$ | $r$ | $l$ | $n$ | $r$ (unchanged) | firing: |
| 0 | 0 | 0 | 0 | 1 | 0 | $t_{13}$ |
| 0 | 0 | 1 | 0 | 0 | 1 | $t_{14}$ |
| 0 | 1 | 0 | 1 | 1 | 0 | $t_{15}$ |
| 0 | 1 | 1 | 1 | 0 | 1 | $t_{16}$ |
| 1 | 0 | 0 | 0 | 1 | 0 | $t_{17}$ |
| 1 | 0 | 1 | 0 | 0 | 1 | $t_{18}$ |
| 1 | 1 | 0 | 1 | 1 | 0 | $t_{19}$ |
| 1 | 1 | 1 | 1 | 0 | 1 | $t_{20}$ |

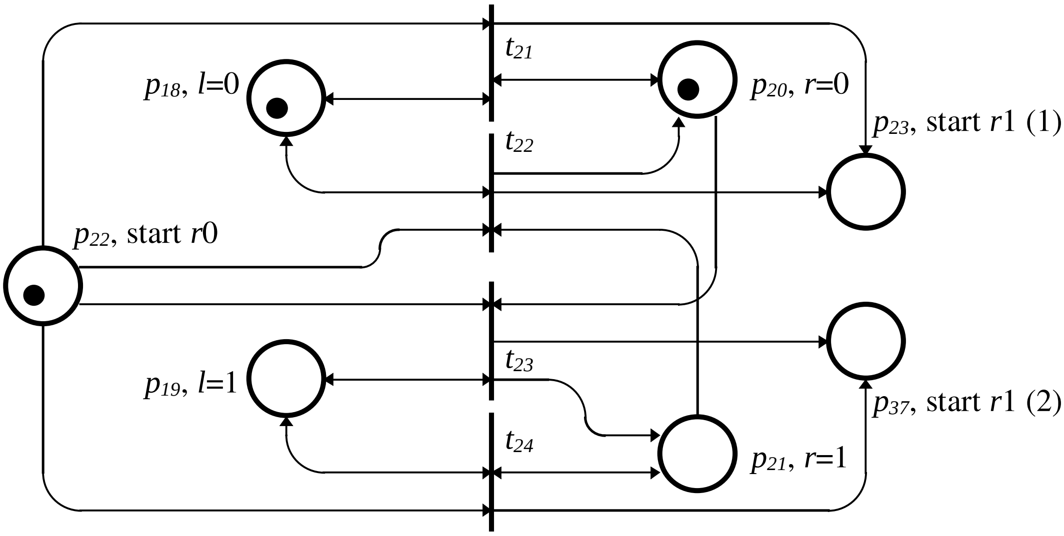


**Figure 16 Petri net model of the four-slot mechanism hardware implementation (Part 4: $r := l$)**

**Listing 15 Petri net model of the four-slot mechanism hardware implementation (4)**

(Part 4. Sub-net representing $r0$: $r := l$ : $\{P_{r0}, T_{r0}, I, O\}$)
structure:

| | |
|---|---|
| $P_{r0} = \{p_i: i = 19\sim23,37\}$ | $T_{r0} = \{t_j: j = 21\sim24\}$ |
| $I(t_{21}) = \{p_{18}, p_{20}, p_{22}\}$ | $O(t_{21}) = \{p_{18}, p_{20}, p_{23}\}$ |
| $I(t_{22}) = \{p_{18}, p_{21}, p_{22}\}$ | $O(t_{22}) = \{p_{18}, p_{20}, p_{23}\}$ |
| $I(t_{23}) = \{p_{19}, p_{20}, p_{22}\}$ | $O(t_{23}) = \{p_{19}, p_{21}, p_{37}\}$ |
| $I(t_{24}) = \{p_{19}, p_{21}, p_{22}\}$ | $O(t_{24}) = \{p_{19}, p_{21}, p_{37}\}$ |
| initial state: $m_{22} = 1$ | $m_{23} = m_{37} = 0$. |
| model: $m_{22}, = 1 \rightarrow$ starting $r0$ | $m_{23}$ or $m_{37}, = 1 \rightarrow$ starting $r1$ |

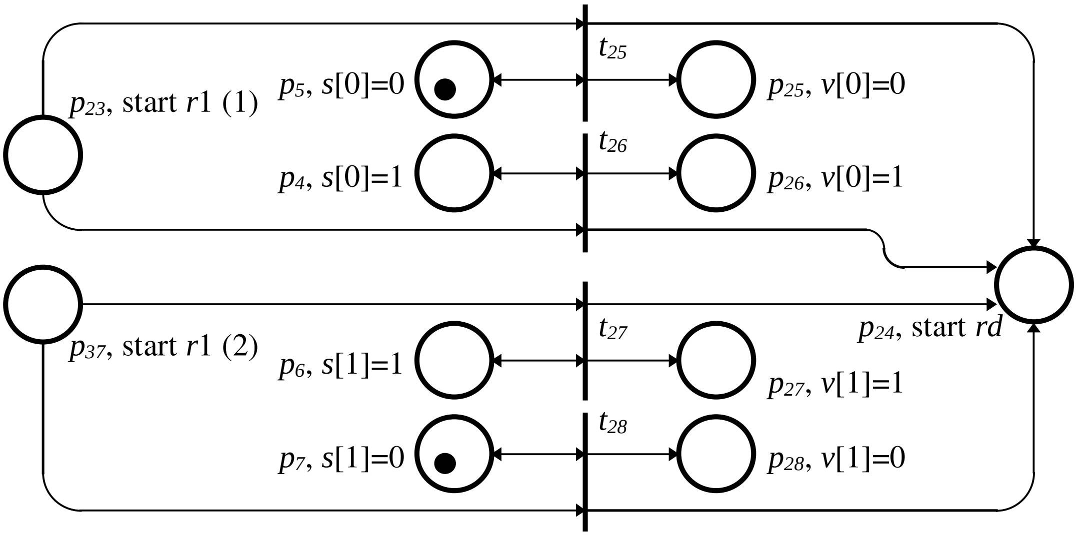


**Figure 17 Petri net model of the four-slot mechanism hardware implementation (Part 5: $v := s$)**

**Listing 16 Petri net model of the four-slot mechanism hardware implementation (5)**

(Part 5. Sub-net representing $r1$: $v := s$ : $\{P_{r1}, T_{r1}, I, O\}$)

structure:

$P_{r1} = \{p_i: i = 4\text{~}7, 23\text{~}28, 37\}$ $\quad$ $T_{r1} = \{t_j: j = 25\text{~}28\}$

| | |
|---|---|
| $I(t_{25}) = \{p_5, p_{23}\}$ | $O(t_{25}) = \{p_5, p_{24}, p_{25}\}$ |
| $I(t_{26}) = \{p_4, p_{23}\}$ | $O(t_{26}) = \{p_4, p_{24}, p_{26}\}$ |
| $I(t_{27}) = \{p_6, p_{37}\}$ | $O(t_{27}) = \{p_6, p_{24}, p_{27}\}$ |
| $I(t_{28}) = \{p_7, p_{37}\}$ | $O(t_{28}) = \{p_7, p_{24}, p_{28}\}$ |
| initial state: | $m_{24} = m_{25} = m_{26} = m_{27} = m_{28} = 0$. |
| model: | $m_{24} = 1 \rightarrow$ start of $rd$ |
| $m_{25} = 1 \rightarrow v[0] = 0$ | $m_{26} = 1 \rightarrow v[0] = 1$ |
| $m_{27} = 1 \rightarrow v[1] = 1$ | $m_{28} = 1 \rightarrow v[1] = 0$ |

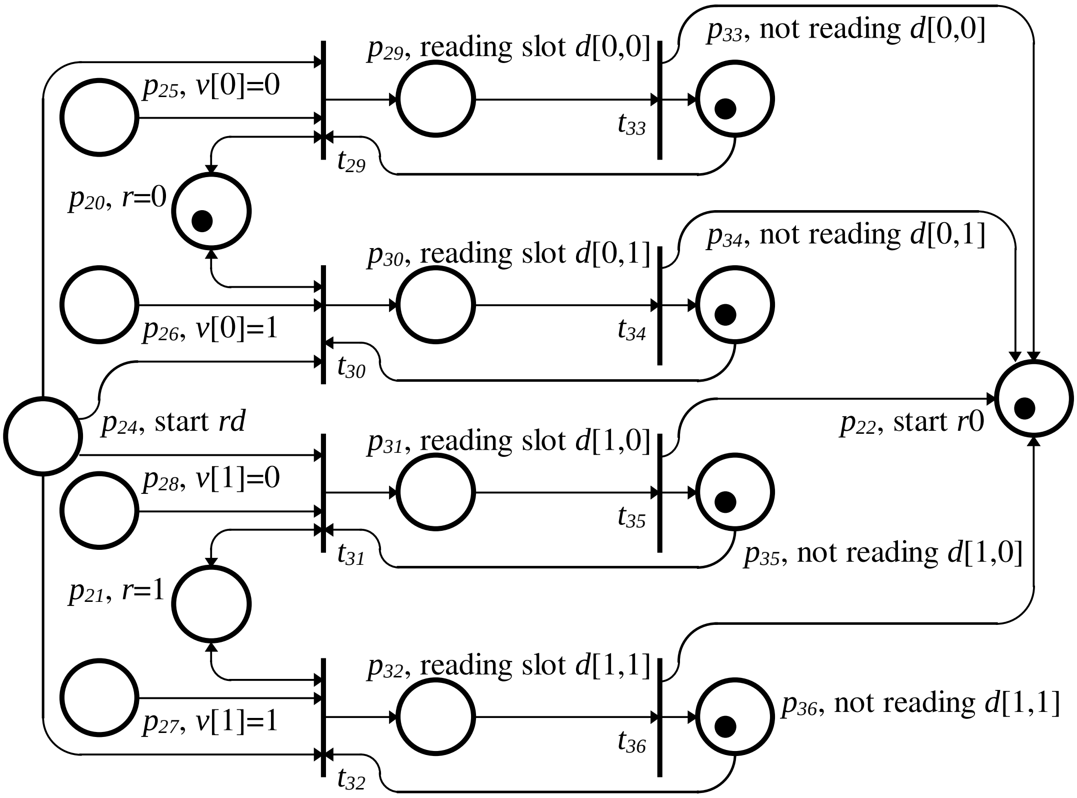


**Figure 18 Petri net model of the four-slot mechanism hardware implementation (Part 6:** $output := d[r, v[r]]$ **)**

**Listing 17 Petri net model of the four-slot mechanism hardware implementation (6)**

(Part 6. Sub-net representing $rd$: $output := d[r, v[r]]$ : $\{P_{rd}, T_{rd}, I, O\}$)

structure:

| | |
|---|---|
| $P_{rd} = \{p_i: i = 20\sim36\}$ | $T_{rd} = \{t_j: j = 29\sim36\}$ |
| $I(t_{29}) = \{p_{20}, p_{24}, p_{25}, p_{33}\}$ | $O(t_{29}) = \{p_{20}, p_{29}\}$ |
| $I(t_{30}) = \{p_{20}, p_{24}, p_{26}, p_{34}\}$ | $O(t_{30}) = \{p_{20}, p_{30}\}$ |
| $I(t_{31}) = \{p_{21}, p_{24}, p_{28}, p_{35}\}$ | $O(t_{31}) = \{p_{21}, p_{31}\}$ |
| $I(t_{32}) = \{p_{21}, p_{24}, p_{27}, p_{36}\}$ | $O(t_{32}) = \{p_{21}, p_{32}\}$ |
| $I(t_{33}) = \{p_{29}\}$ | $O(t_{33}) = \{p_{22}, p_{33}\}$ |
| $I(t_{34}) = \{p_{30}\}$ | $O(t_{34}) = \{p_{22}, p_{34}\}$ |
| $I(t_{35}) = \{p_{31}\}$ | $O(t_{35}) = \{p_{22}, p_{35}\}$ |
| $I(t_{36}) = \{p_{32}\}$ | $O(t_{36}) = \{p_{22}, p_{36}\}$ |

initial state:

| | |
|---|---|
| $m_{33} = m_{34} = m_{35} = m_{36} = 1$ | $m_{29} = m_{30} = m_{31} = m_{32} = 0.$ |

model:

| | |
|---|---|
| $m_{29} = 1 \rightarrow$ reading slot $d[0,0]$ | $m_{30} = 1 \rightarrow$ reading $d[0,1]$ |
| $m_{31} = 1 \rightarrow$ reading $d[1,0]$ | $m_{32} = 1 \rightarrow$ reading $d[1,1]$ |
| $m_{33} = 1 \rightarrow$ not reading $d[0,0]$ | $m_{34} = 1 \rightarrow$ not reading $d[0,1]$ |
| $m_{35} = 1 \rightarrow$ not reading $d[1,0]$ | $m_{36} = 1 \rightarrow$ not reading $d[1,1]$ |

## 1.1 References


1 Simpson, H.R., “Four-slot fully asynchronous communication mechanism”, *IEE Proceedings*, Vol. 137, Pt. E, No. 1, pp.17-30, January 1990.

2 Simpson, H.R., “Correctness analysis of class of asynchronous communication mechanisms”, *IEE Proceedings*, Vol. 139, Pt. E, No. 1, pp.35-49, January 1992.

3 “Asynchronous communication mechanisms”, European Patent, No. 0 292 287, 1988.

4 Peterson, J.L., *Petri net theory and the modeling of systems*, Prentice-Hall, 1981.

5 Merlin, P.M. “A methodology for the design and implementation of communication protocols”, *IEEE Trans. Commun.*, Vol. COM-24, No. 6, pp.614-621, June 1976.

6 Molloy, M.K., “Performance analysis using stochastic Petri nets”, *IEEE Trans. Comput.*, Vol. C-31, pp.913-917, Sept. 1982.

7 Vallejo, Gregorio, J.A., Gonzalez, M. and Darke, J.M., “Shared memory multiprocessor operating system with an extended Petri net model”, *IEEE Trans. Paral. & Dist. Syst.*, Vol. 5, No. 7, July 1994.

8 Li, M., and Georganas, N.D., “Algorithms to analyse Petri nets: exact parametric analysis of stochastic Petri nets”, *IEEE Trans Comput*., Vol. 41, No. 9, pp.1176-1180, Sept. 1992.

9 Chen, Y., Tsai, W.T. and Chao, D., “Dependency analysis - a Petri net based technique for synthesizing large concurrent systems”, *IEEE Trans. Paral. & Dist Syst.*, Vol. 4, No. 4, pp.414-426, April 1993.