УДК 007.51

В. И. Волчихин, Д. В. Пащенко, Д. А. Трокоз

МОДЕЛИРОВАНИЕ ПОДСИСТЕМЫ ЗАГРУЗКИ ДАННЫХ НАЗЕМНОЙ СИСТЕМЫ КОНТРОЛЯ АВИАЦИОННЫХ РАДИОЛОКАЦИОННЫХ КОМПЛЕКСОВ С ИСПОЛЬЗОВАНИЕМ АППАРАТА СЕТЕЙ ПЕТРИ

Аннотация. Обсуждаются вопросы моделирования подсистемы загрузки данных наземной системы контроля для авиационных радиолокационных комплексов с использованием сетей Петри. С использованием результатов моделирования производится выбор аппаратных средств системы контроля.

Ключевые слова: радиолокационный комплекс, анализ систем, система объективного контроля, сети Петри.

Abstract. Discusses the simulation subsystem data download ground control systems for aircraft radar systems using Petri nets. The results of simulation used to select the hardware monitoring systems.

Keywords: radar system, systems analysis, system of objective control, Petri nets.

Введение

Перспективные авиационные комплексы радиолокационного дозора и наведения (АК РЛДН) для оценки качества выполнения полетных заданий и технического состояния радиотехнических комплексов (РТК) используют экспертные системы, которые осуществляют объективный контроль. Объем обрабатываемой информации в этих системах может достигать нескольких терабайт [1, 2]. При этом в соответствии с авиационными правилами производства полетов государственной авиации [3] объективный контроль подразделяется на оперативный, специальный и полный. Более трудоемкими являются специальный и полный контроли [4]. Для проведения таких видов контроля требуются специальные вычислительные ресурсы с возможностью параллельной обработки данных [5]. Для определения структуры и состава таких ресурсов необходимо произвести моделирование работы вычислительного комплекса с использованием алгоритмов работы системы объективного контроля радиолокационных комплексов воздушного базирования.

1. Принципы построения модели

Наиболее длительными и трудоемкими процессами при работе системы объективного контроля являются загрузка данных с бортового устройства регистрации в базу данных и автоматизированный контроль аппаратуры и выполнения полетного задания [6]. В данной статье рассматривается модель загрузки данных в систему объективного контроля АК РЛДН с использованием механизма сетей Петри.

При построении однопоточной модели на сетях Петри позиции будут выступать в роли элементов хранения данных, которые представлены фишками, а переходы – в роли процессов или действий, осуществляемых над данными (фишками). При этом каждая фишка будет относиться к определенному множеству цветов – типу данных.

Обработка данных в модели происходит при переходе, т.е. перемещении фишки из одной позиции в другую. Она заключается в изменении цветов

входных фишек (поступающих на вход перехода), а также непропускании на выход некоторых входных фишек и создании новых выходных фишек. Это позволяет моделировать процесс преобразования данных для любой однопоточной системы.

При создании многопоточной системы необходимо ввести специальный тип фишек, который представляет потоки в модели. Это фишки множества цветов целых чисел, обладающих свойством времени:

colset Thread = StructThread timed,

где StructThread – структура, содержащая номер и информационные поля потока.

В многопоточной модели лишь множество цветов потоков должно обладать свойством времени. При этом каждый тип данных, который участвует в многопоточной обработке, будет преобразован в следующую структуру:

colset TYPE_T = record thread:Thread*data:TYPE,

где TYPE – базовый тип данных, который будет участвовать в многопоточной обработке; Thread – поток, который производит обработку данных.

Используя такое представление, получаем два вида множеств цветов: первый – типы данных, второй – состоящий из одного специального множества, представляющего потоки модели. В многопоточной модели для каждой позиции, хранящей фишки, для нескольких потоков создается позиция, хранящая потоки. Переход будет разрешен, если входные фишки данных соответствуют входной фишке потока. Проследить работу модели можно, наблюдая за изменением кортежа, содержащего количество фишек потоков. При этом потоки влияют лишь на те данные, которые им разрешено обрабатывать, исходя из топологии модели.

Для реализации функционирования модели необходимо для каждого перехода задать следующие условия:

(vThread = #thread x_1),

andalso (vThread = #thread x_2),

...

andalso (vThread = #thread x_n),

где x_i – i-е имя входной переменной; n – число входных переменных.

Множество условий переходов обеспечит соответствие фишек-данных и фишек-потоков, что обеспечит независимую обработку данных, относящуюся к разным потокам.

В любой системе с многопоточностью существуют этапы обработки, которые в единый момент времени могут производиться лишь одним из потоков (например, чтение данных с диска). Для построения модели с многопоточной обработкой требуется ввод в модель критической секции – участка сети Петри, в котором единовременно может находиться лишь один поток (в общем случае ограниченное количество потоков).

Для реализации критической секции вводятся две позиции:

– Free (критическая секция свободна);

– Take (критическая секция занята).

В добавление к позициям вводится особое множество цветов Mark, которое содержит единственный цвет:

colset Mark = with mark.

Число фишек типа Mark в позиции Free в начальной расстановке фишек указывает на число критических ресурсов, т.е. число потоков, которые могут единовременно находиться в критической секции. Пока в позиции Free есть фишки, в критическую секцию могут поступать потоки. При поступлении очередного потока в критическую секцию одна фишка Mark переходит из позиции Free в позицию Take. Когда поток выходит из критической секции, т.е. освобождает критический ресурс, одна фишка Mark из позиции Take переходит в позицию Free.

Пример реализации многопоточной обработки данных с использованием сети Петри в визуальном редакторе CPN Tools представлен на рис. 1.

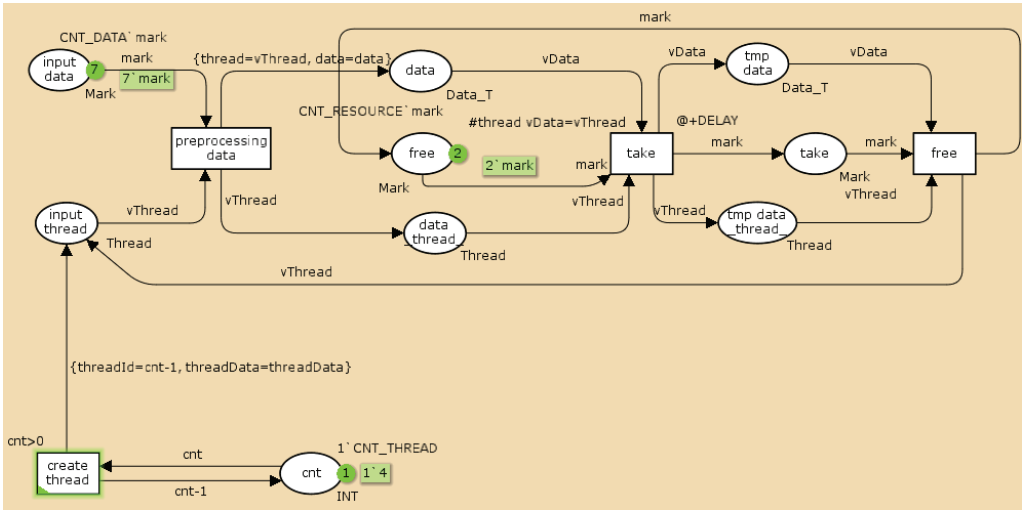

Рис. 1. Пример реализации многопоточной обработки данных с использованием сети Петри

Входная информация поступает в переход preprocessing data одновременно с обрабатывающим ее потоком. При этом происходит сопоставление потока и данных. Поскольку количество потоков может быть больше, чем критических ресурсов, то в модели вводится критическая секция. Критическая секция реализуется с помощью переходов и позиций Take и Free, как описано выше.

Подобный подход позволяет строить модели многопоточных систем, имеющих критические ресурсы.

При построении модели многопоточной системы, какой является система загрузки наземного комплекса обработки и дешифрирования информации (НКОД), позиции вводятся парами: одна позиция хранит фишки-данные, а другая – фишки-потоки. Поэтому вводим две позиции:

files – позиция хранит фишки, представляющие собой входные файлы;

threads – позиция хранит фишки, представляющие собой потоки для обработки входных файлов.

2. Модели работы системы контроля

Модель НКОД представляет собой трехуровневую сеть Петри, первый уровень которой представлен на рис. 2.

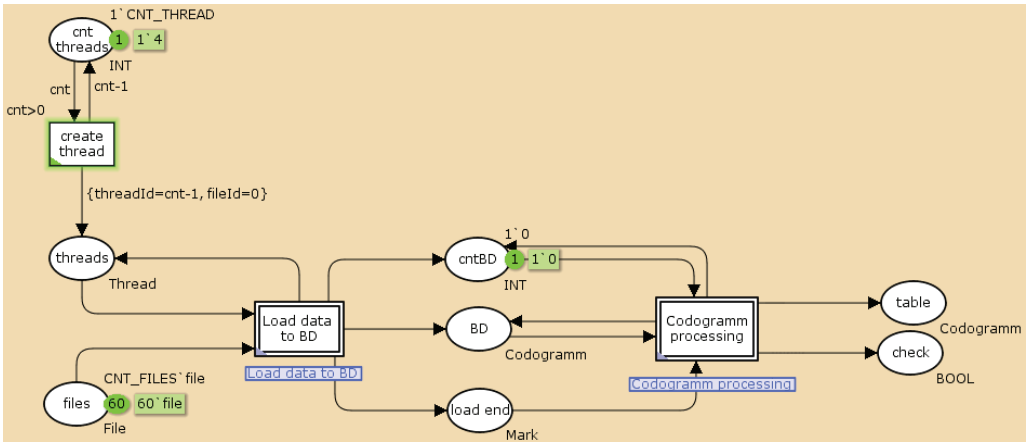

Рис. 2. Первый уровень модели НКОД, реализованный в CPN tools

В начале функционирования сети происходит генерация потоков, которые в виде фишек попадают в позицию threads.

Переход Load data to BD моделирует загрузку данных в систему объективного контроля и является переходом на второй уровень модели НКОД, который представлен на рис. 3. Из позиций thread и files фишки потоков и входных данных обрабатываются переходом Read file size and header, который моделирует процесс чтения размера файла и его заголовка. Этот переход создает переменные (фишки), которые необходимы для дальнейшей обработки файла и помещает их в следующие позиции:

File end – позиция хранит фишки типа Bool_T, которые способны принимать значения «ложь» и «истина» и являются признаком окончания чтения файла, если фишка примет истинное значение. В эту позицию при загрузке очередного файла помещается фишка с ложным значением, которая символизирует, что процесс чтения файла не был окончен.

Save header – позиция хранит фишки типа HeaderCod_T, этот тип хранит заголовок кодограммы (номер кодограммы и размер). Эта позиция используется для хранения заголовков кодограммы, если возникает ситуация, когда при чтении тела кодограммы из буфера выясняется, что оно полностью не поместилось в оставшуюся часть буфера и необходимо произвести очередное чтение из файла в буфер, но заголовок кодограммы уже был прочитан из буфера, поэтому его нужно сохранить в эту позицию, прочитать новую порцию данных из файла в буфер и начать чтение из буфера не с заголовка кодограммы, а с ее тела, используя сохраненный заголовок. Эта позиция при загрузке очередного файла заполняется фишкой, содержащей нулевые значения номера и размера кодограммы, т.е. просто создается переменная, ассоциированная с потоком, читающим файл, которая в будущем будет использована для хранения заголовка.

IsSave header – позиция хранит фишки типа BOOL_T, которые являются признаком хранения заголовка кодограммы при обработке файла соответствующим потоком.

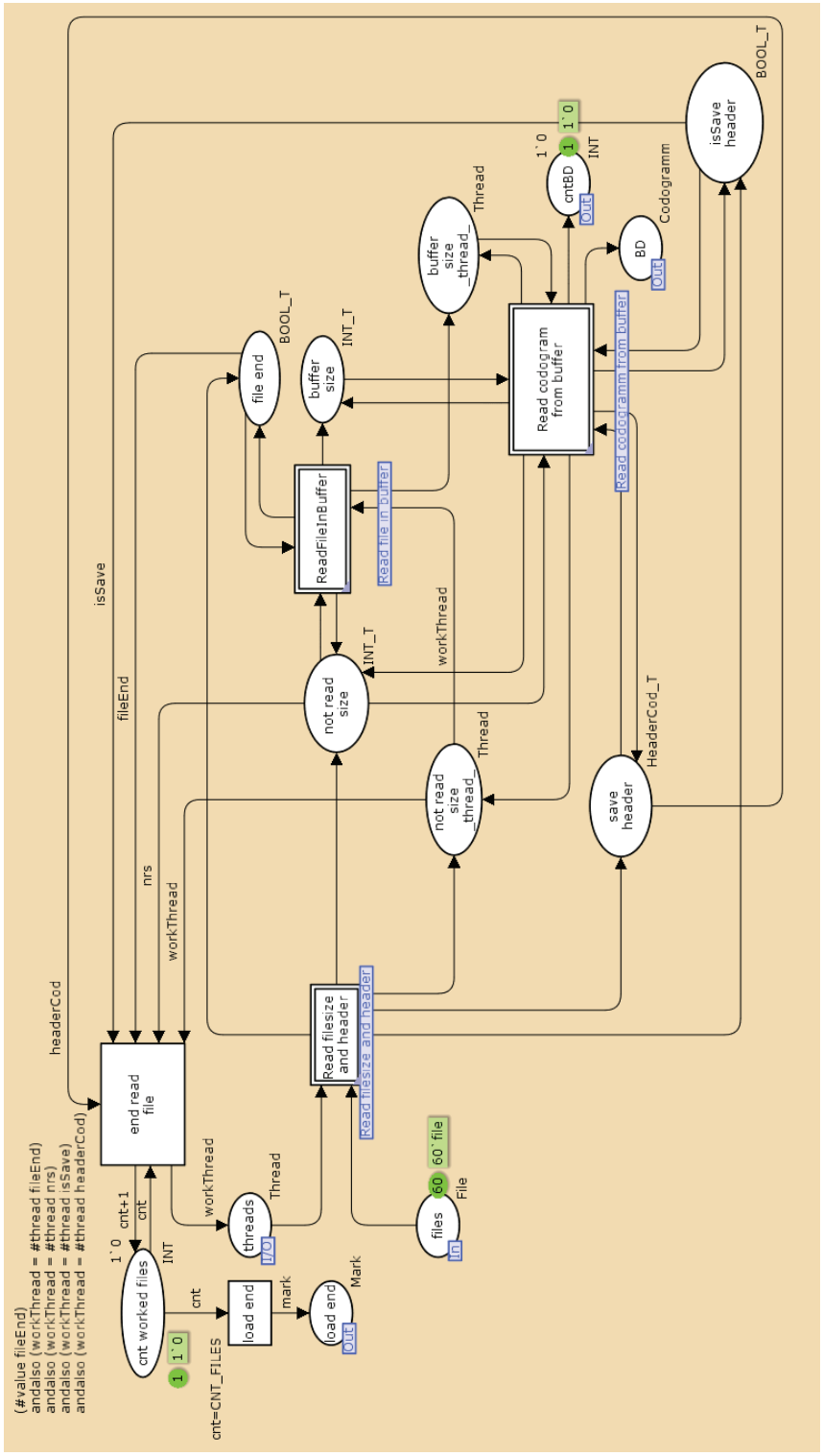

Рис. 3. Второй уровень модели НКОД (Load data to BD)

Эта позиция при загрузке очередного файла заполняется ложным значением и используется вместе с фишкой в позиции «Save header», которая непосредственно хранит сам заголовок кодограммы. Если заголовок будет сохранен соответствующим потоком, то соответствующая сохранившему заголовок потоку фишка в позиции «isSave header» примет значение «истина».

Not read size – позиция хранит фишки типа INT_T, содержащие размер непрочитанной части файла. Эта позиция при загрузке очередного файла заполняется значением размера файла.

not read size _thread_ – позиция хранит фишки типа Thread, содержащие потоки, связанные с позицией «not read size». Эта позиция заполняется текущим рабочим потоком.

Переход Read file in buffer является переходом на третий уровень модели и производит чтение из файла в буфер с учетом использования диска как критического ресурса (рис. 4).

Переходы, моделирующие процесс чтения, содержатся в двух ветвях сети: одна – для чтения файла в буфер, когда размер оставшейся части файла больше размера буфера, а другая – когда меньше, т.е. вторая ветвь необходима для последнего чтения из файла и установки фишки-флага, указывающего на конец файла в позиции file end, в истинное значение.

При срабатывании перехода take hdd and read или last take hdd and read происходит перемещение фишек из позиций not read size, not read size _thread_ и free hdd в позиции nrs, nrs _thread_ и take hdd, т.е. происходит захват критического ресурса (жесткого диска). Занятость критического ресурса показывают следующие позиции:

free hdd – позиция хранит фишки типа Mark; если фишка присутствует в данной позиции, значит, критический ресурс свободен;

take hdd – позиция хранит фишки типа Mark; если фишка присутствует в данной позиции, значит, критический ресурс занят.

Переходы take hdd and read и last take hdd and read обладают временной задержкой, которая соответствует времени чтения файлов с диска. Данная задержка формируется в зависимости от размера читаемых данных и скорости чтения с диска. Она определяется по одному из следующих законов:

– BUFFER_SIZE * READ_TIME – когда размер непрочитанных данных больше или равен буферу для чтения;

– #value nrs * READ_TIME – когда размер непрочитанных данных меньше буфера для чтения,

где #value nrs – размер непрочитанных данных из файла; BUFFER_SIZE – размер буфера для чтения; READ_TIME – среднее время чтения одного байта данных.

Далее срабатывают переходы, которые освобождают критический ресурс:

free HDD – переход, осуществляющий чтение из файла в буфер обмена и освобождение критического ресурса, когда размер оставшейся части файла больше размера буфера;

last free HDD – переход, осуществляющий чтение из файла в буфер обмена и освобождение критического ресурса, когда размер оставшейся части файла меньше или равен размеру буфера.

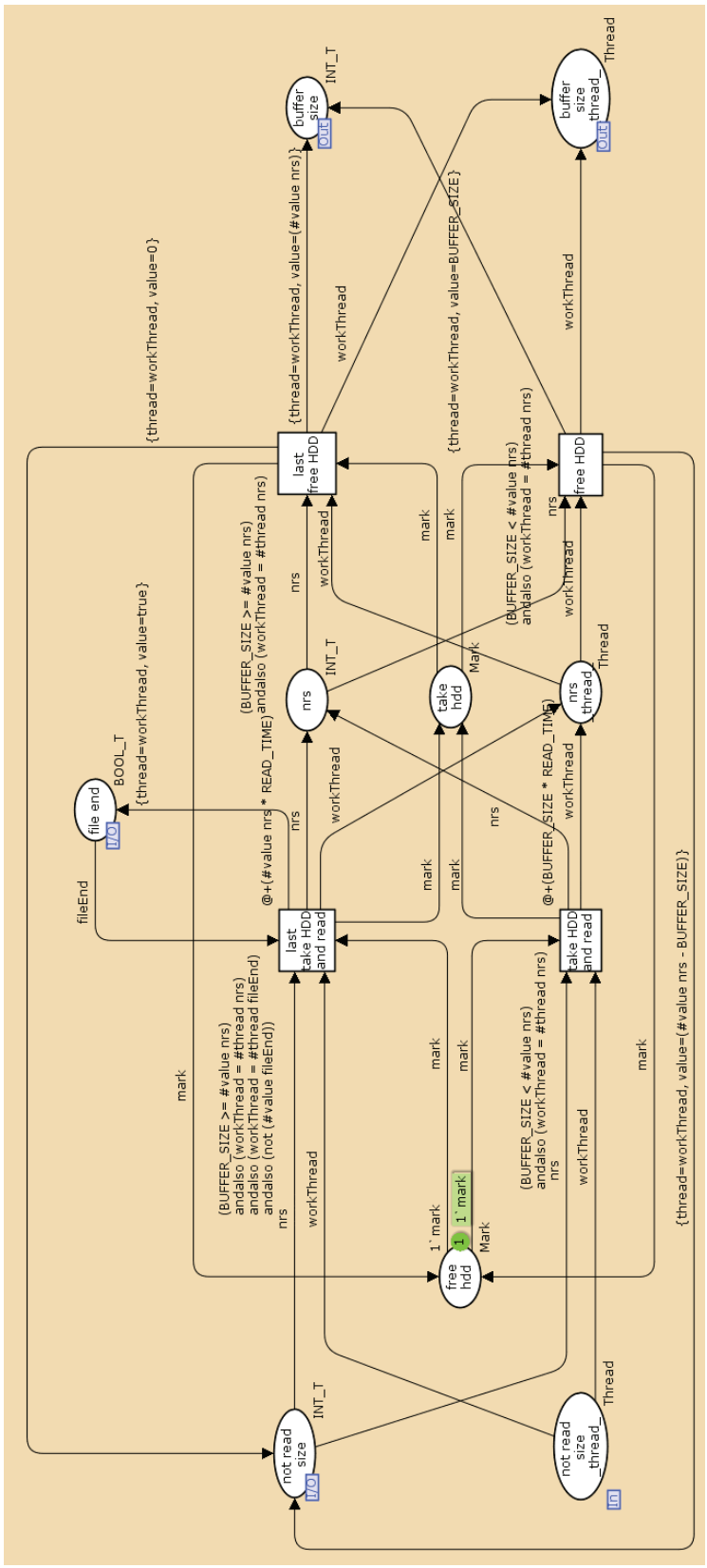

Рис. 4. Третий уровень модели НКОД (*Read file in buffer*)

При срабатывании первого перехода в позицию not read size помещается фишка, имеющая значение, равное «not read size – BUFFER_SIZE», т.е. объем непрочитанных данных из файла уменьшается на величину буфера данных. Если же срабатывает второй переход, т.е. происходит последнее чтение данных, то в позицию not read size помещается фишка, имеющая значение, равное нулю. Кроме того, при срабатывании любого из переходов происходит освобождение критического ресурса и передача фишки, содержащей объем прочитанных данных, в позицию buffer size, а также передача рабочего потока в позицию buffer size _thread_.

Выходные данные перехода Read file in buffer, который производит чтение из файла в буфер, являются входными для перехода Read codogramm from buffer, осуществляющего процесс чтения кодограмм из буфера. Он является переходом на третий уровень модели и производит чтение кодограмм из буфера в базу данных (рис. 5).

Переход read codogramm header осуществляет чтение заголовка кодограммы из буфера. При этом значение фишки из позиции buffer size, поступающей на вход перехода, уменьшается на величину, равную длине заголовка кодограммы, и помещается обратно в позицию buffer size. Прочитанный заголовок кодограммы помещается в виде фишки типа HeaderCod_T в позицию codogramm header, а рабочий поток из позиции buffer size_thread_ – в позицию codogramm header_thread_. Этот переход срабатывает, если ранее не был сохранен заголовок кодограммы, т.е. если флаг в позиции isSave header сохранения заголовка кодограммы для данного рабочего потока ложный.

Переход load header срабатывает, если заголовок кодограммы был прочитан ранее. Он производит перенос фишки, содержащей сохраненный заголовок кодограммы в позицию codogramm header, при этом в позиции isSave header соответствующая фишка-флаг сбрасывается.

После того как заголовок кодограммы был помещен в позицию codogramm header и, соответственно, был получен размер тела кодограммы, можно ее прочитать. Переход read codogramm производит чтение тела кодограммы. Ему на вход поступают фишки, содержащие заголовок кодограммы, размер буфера и рабочий поток, в результате его срабатывания происходит чтение кодограммы в БД, а на выходе формируется фишка, содержащая размер оставшихся данных в буфере, и фишка – рабочий поток, которые помещаются в позиции buffer size и buffer size_thread_ соответственно.

Переходы end buffer read header и end buffer read codogramm отслеживают окончание данных в буфере:

end buffer read header – переход срабатывает, если при попытке прочитать очередной заголовок кодограммы выяснилось, что объем данных в буфере меньше длины заголовка кодограммы. При этом происходит добавление длины данных, оставшихся в буфере buffer size, к длине непрочитанных данных, хранящихся в фишке в позиции not read size, и помещение результирующей фишки в позицию not read size. Кроме того, рабочий поток помещается из позиции buffer size_thread_ в позицию not read size_thread_. На этом работа вложенного перехода чтения данных из файла в буфер завершается;

end buffer read codogramm – переход срабатывает, если при попытке прочитать очередное тело кодограммы выяснилось, что объем данных в буфере меньше длины тела кодограммы.

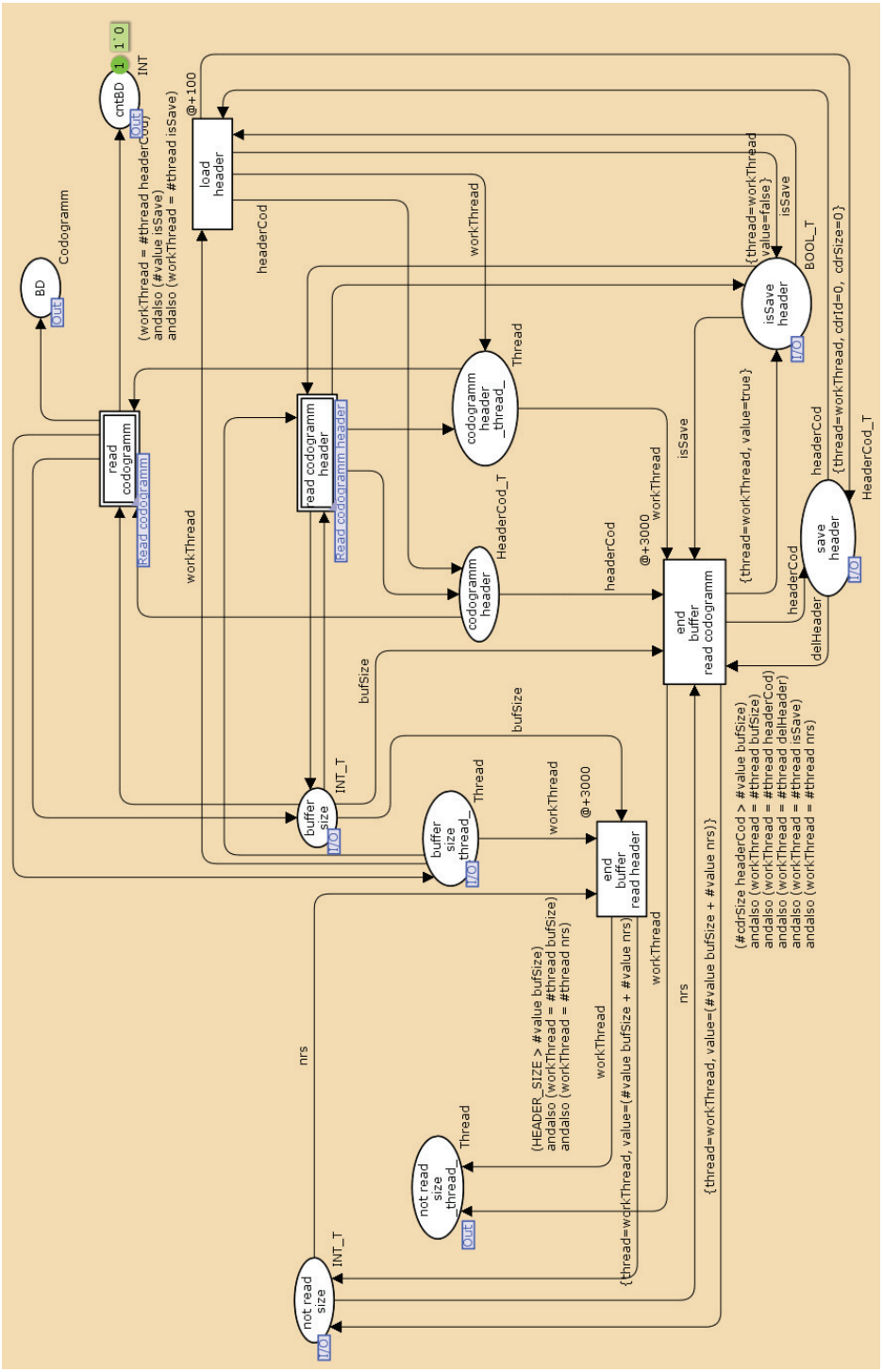

Рис. 5. Третий уровень модели НКОД (Read codogramm from buffer)

При этом происходит добавление длины данных, оставшихся в буфере buffer size, к длине непрочитанных данных, хранящихся в фишке в позиции not read size, и помещение результирующей фишки в позицию not read size. Так же происходит сохранение заголовка кодограммы путем помещения фишки-заголовка кодограммы из позиции codogramm header в позицию save header и установления флага в позиции isSave header. Кроме того, рабочий поток помещается из позиции buffer size_thread_ в позицию not read size_thread_.

Переход end read file срабатывает, когда флаг окончания файла, хранимый в позиции file end, принимает истинное значение. При этом фишки из позиций not read size, save header и isSave header, поступая на вход перехода, уничтожаются, а фишка потока из позиции not read size_thread_ помещается в позицию threads. Таким образом, поток становится доступным для чтения следующего файла из позиции files. Работа перехода Load data to BD завершается, если в позиции files не остается файлов, а в позицию threads возвращаются все потоки, т.е. в том случае, если файлов, ожидающих обработки, не осталось и все потоки завершили свою работу.

С использованием построенной иерархической сети Петри было проведено моделирование. Время данных, обработанных на имитаторе, представлено в табл. 1.

Таблица 1

Расчетное время (с)

Число процессоров	RAID0(2)	RAID10(4)	RAID10(6)
1	10,8	9,5	9,0
2	6,9	5,6	5,1
4	4,9	3,9	3,3
8	4,3	3,1	2,9
16	4,2	3,0	2,8

3. Результаты моделирования

Эмпирическим путем с помощью данных имитатора наземной системы контроля было проанализировано время работы системы с известными параметрами дисковой подсистемы и системы обработки.

Результаты сравнительной оценки построения систем ввода-вывода, полученные в результате моделирования, представлены в табл. 1 и на рис. 6.

В табл. 1 представлено время обработки дисковой подсистемой 1 % от объема, предполагаемого к обработке на реальной системе. Данная модель строилась на основе данных, полученных на имитаторе системы контроля, построенном на основе данных реальных полетов самолета дальнего радиолокационного обнаружения и управления А-50. Исследования свойств этого имитатора, а также структуры и состава входных данных не входят в круг вопросов, рассматриваемых в данной статье, и приводятся as..is (как есть).

Заключение

Используя полученные данные обработки, зная объемы обрабатываемой информации, можно сделать выбор в пользу необходимой конфигурации для системы наземного контроля авиационных радиолокационных комплексов.

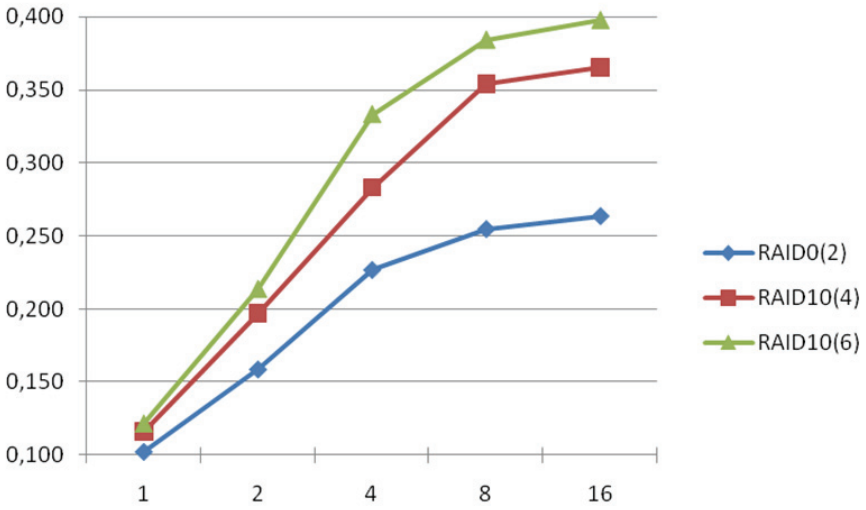

Рис. 6. График зависимости производительности системы контроля от количества потоков обработки (1–16) и дисковой подсистемы (Raid)

Данная система планируется к применению при проектировании корабельной обработки данных объективного контроля АК РЛДН авианосного базирования.

Список литературы

1. **Волчихин, В. И.** Экспертная система контроля и документирования в авиации / В. И. Волчихин, Д. В. Пащенко, А. Н. Токарев, Н. Н. Коннов // Автоматизация и управление в технических системах : межвузовский сборник научных трудов. – Пенза : Информационно-издательский центр ПГУ, 2008. – С. 170–175.
2. **Пащенко, Д. В.** Система для расшифровки и анализа данных объективного контроля радиолокационного комплекса / Н. Н. Коннов, Д. В. Пащенко, Г. М. Морозов, А. В. Васильев // Новые информационные технологии и системы : труды 8 Международной научно-технической конференции. – Пенза, 2008. – С. 274–281.
3. Приказ министра обороны РФ от 24.09.2004 № 275 об утверждении федеральных авиационных правил производства полетов государственной авиации (зарегистрировано в Минюсте РФ 10.11.2004 № 6110) // Законодательство РФ. – 2004. – Ч. 1. – URL: http://2004-1.xof.ru/lib/?tm=151&vp=akt15155 (Дата обращения 23.09.2010).
4. **Пащенко, Д. В.** Объективный контроль состояния авиационных радиолокационных комплексов / Д. В. Пащенко // Проблемы автоматизации и управления : труды Международной научно-технической конференции. – Пенза, 2009. – С. 55–59.
5. **Трокоз, Д. А.** Стратегия параллельной обработки массивов данных в системе объективного контроля радиотехнического комплекса радиолокационного дозора и наведения / В. А. Мачалин, Д. В. Пащенко, Д. А. Трокоз, М. Н. Синев // Вопросы радиоэлектроники. – 2009. – Вып. 4. – С. 139–144. – (Сер. ЭВТ).
6. **Пащенко, Д. В.** Методика построения систем объективного контроля авиационных радиолокационных комплексов / Д. В. Пащенко, М. Н. Синев // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2009. – № 4. – С. 49–59.

Волчихин Владимир Иванович
доктор технических наук, профессор, ректор Пензенского государственного университета

E-mail: rektor@pnzgu.ru

Volchikhin Vladimir Ivanovich
Doctor of engineering sciences, professor, rector of Penza State University

Пащенко Дмитрий Владимирович
кандидат технических наук, доцент, кафедра вычислительной техники, Пензенский государственный университет

E-mail: Dmitry.pashchenko@gmail.com

Pashchenko Dmitry Vladimirovich
Candidate of engineering sciences, associate professor, sun-department of computer engineering, Penza State University

Трокоз Дмитрий Анатольевич
магистрант, Пензенский государственный университет

E-mail: Dmitriy.trokoz@gmail.com

Trokoz Dmitry Anatolyevich
Undergraduate student, Penza State University

УДК 007.51
Волчихин, В. И.

Моделирование подсистемы загрузки данных наземной системы контроля авиационных радиолокационных комплексов с использованием аппарата сетей Петри / В. И. Волчихин, Д. В. Пащенко, Д. А. Трокоз // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2010. – № 3 (15). – С. 37–48.