УДК 681.325.5: 518.5

ФОРМАЛИЗМ АДРЕСНО–ВРЕМЕННЫХ КАРТ ДЛЯ ОПИСАНИЯ АЛГОРИТМОВ ФУНКЦИОНИРОВАНИЯ МНОГОКАНАЛЬНЫХ СИСТЕМ УПРАВЛЕНИЯ

Введение в формализм адресно–временных карт

А. М. Астапкович,
канд. техн. наук, начальник СКБ
Санкт-Петербургский государственный университет аэрокосмического приборостроения (ГУАП)

Работа посвящена изложению нового способа укрупненного описания алгоритмов функционирования многоканальных систем управления реального времен. Предложенный формализм включает в себя понятия: процессограмма, циклограмма, адресно-временные карты. Рассматриваются алгоритмические аспекты использования предложенного способа описания и символьное представление алгоритмов для наиболее распространенных дисциплин диспетчеризации. Вводятся три типа конфликтов в глобальном конкурентном пространстве.

Article concerns the new method of the robust description of the multichannel real-time control system algorithm. Proposed formalism includes: processogramm, cyclogramm, address-time card. Algorithm aspects were examined also as symbolic description for the most popular scheduler disciplines. Three types of the conflicts in the global concurrent space were introduced

Введение

В связи с бурным развитием встраиваемых систем управления проблема существенного повышения скорости разработки прикладного программного обеспечения (ППО) для них приобретает особую актуальность. Особый интерес представляют многоканальные микропроцессорные системы управления реального времени встраиваемого класса, для которых принципиально важной является реализация предписанного алгоритма управления по каждому из каналов за время, не превышающее специфицированного. Высоконадежные многоканальные системы управления, обрабатывающие асинхронные входные воздействия, относят к подклассу так называемых «систем жесткого реального времени».

Характерной особенностью таких систем является их «закрытость», т. е. неизменяемость программного обеспечения в нормальных режимах эксплуатации. Для такого класса систем управления в качестве структурной основы прикладного программного обеспечения, как правило, используют специализированные операционные системы реального времени (ОСРВ) [1, 2]. Однако наблюдается тенденция все более широкого использования микрооперационных систем реального времени (mОСРВ) [3], представляющих собой узкоспециализированные ОСРВ для конкретного класса микропроцессорных устройств. С учетом этого проблема автоматизации системной интеграции применительно к многоканальным системам управления реального времени в настоящее время выдвигается на первый план. Решение задачи системной интеграции требует обеспечения бесконфликтного разделения ресурсов между параллельно выполняющимися задачами, а также обеспечения специфицированных времен реакции по каждому из каналов на всем допустимом множестве входных воздействий на объект управления. Вторая по важности задача заключается в необходимости, с одной стороны, обеспечить мобильность разрабатываемого программного обеспечения, а с другой — по возможности автоматизировать процесс интеграции.

Критически взглянув на сложившееся положение в области разработки ППО систем реального времени, можно констатировать, что современные

методы в большей степени фокусируются на разработке собственно логической структуры алгоритма управления, а временной характер задач и их взаимное влияние остаются в тени. Позаимствовав из прикладной математики классификацию «прямая задача — обратная задача», будем рассматривать разработку ППО по традиционной технологии как решение прямой задачи. Причем ее решение осуществляется фактически подбором как с точки зрения логической структуры алгоритма, так и обеспечения временных параметров системы управления.

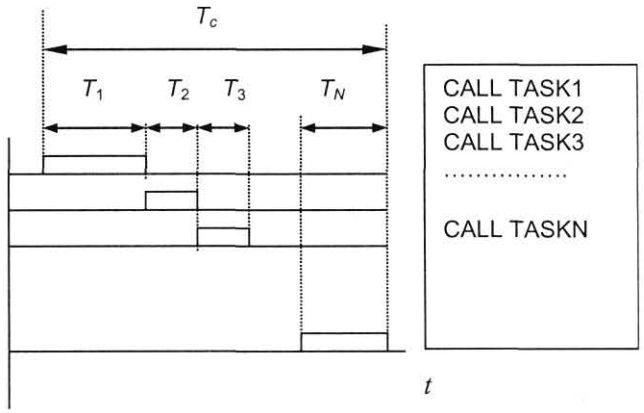
Для многоканальных систем управления, в силу сложного характера информационного взаимодействия, детальное описание алгоритма функционирования становится громоздким из-за необходимости рассматривать поведение системы при широком спектре входных воздействий на длительных интервалах времени. Известна методика применения аппарата имитационного моделирования в сочетании с применением формализма сетей Петри [4, 5], которая в большей степени ориентирована на решение задач анализа логической структуры алгоритмов с использованием технологии типа «прямая задача». Иногда такой подход называют «синтезом через анализ». Сети Петри в этом случае используются как универсальный способ описания структуры исследуемых алгоритмов, а аппарат имитационного моделирования обеспечивает выработку входных воздействий и обработку полученных результатов.

Отсутствие должного внимания к временному аспекту логики работы алгоритма резко ограничивает возможность применения данного аппарата к многоканальным системам управления реального времени. При ближайшем рассмотрении оказывается, что переход к концептуально новым подходам упирается в проблему описаний алгоритмов функционирования, приспособленных к специфике многоканальных систем управления реального времени, когда необходимо некоторое укрупненное описание, позволяющее проводить анализ наличия конфликтов в глобальном конкурентном пространстве для многозадачной ОСРВ. Если пользоваться устоявшейся терминологией, то речь идет о задаче анализа критической секции.

В статье сделана попытка введения математического аппарата, предназначенного для укрупненного и формально строгого описания алгоритмов функционирования многоканальных систем управления реального времени с учетом доступных ресурсов (память, время реализации и т. п.), исходя из общесистемных требований.

Процессограмма и циклограмма

В работе [3] было введено понятие «процессограмма», которое в явном виде отображает параллельный характер процессов. Процессограмма является естественным расширением средств описания алгоритмов функционирования устройств, применяемых до настоящего времени.



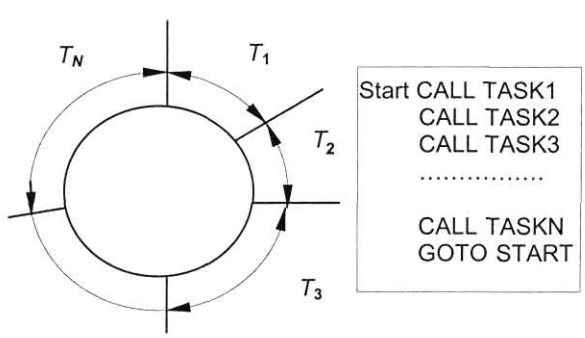
■ ***Рис. 1.*** *Процессограмма для диспетчера FIFO (бесприоритетная линейная очередь с обслуживанием в порядке поступления)*

Иллюстрацией этого понятия является рис. 1, где каждая из задач имеет свою длительность T_i. Структура алгоритма диспетчеризации предельно проста: управление последовательно передается от подпрограммы к подпрограмме. Если принять, что каждая из подпрограмм обеспечивает управление по одному из каналов, то можно рассматривать бесконечный цикл как вариант реализации бесприоритетного циклического диспетчера. В терминах ОСРВ такую диспетчеризацию иногда называют «ядром с кооперативным взаимодействием задач» [1]. Графически эта дисциплина диспетчеризации лучше всего отображается в виде круговой диаграммы, которую будем называть циклограммой (рис. 2).

Процессограмму можно рассматривать как описание одного цикла циклограммы, следовательно, это взаимно дополняющие друг друга понятия. Время опроса T_c всей очереди задач в соответствии с рис. 1 и 2 описывается соотношением

$$T_C = \sum_{i=1}^{N_t} T_i. \qquad (1)$$

Как следует из рис. 1 и 2 и формулы (1), для диспетчеризации с кооперативным взаимодействием задач параллельная обработка входных воздействий обеспечивается при условии, что время



■ ***Рис. 2.*** *Иллюстрация понятия «циклограмма»*

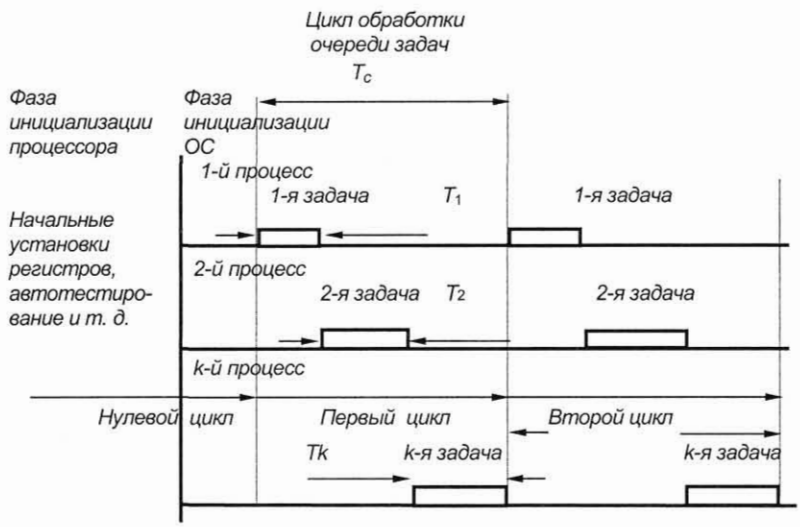
обработки всей очереди задач T_c будет меньше, чем минимальное специфицированное время реакции для всего множества N_t контролируемых каналов. В случае однопроцессорных систем управления речь идет о псевдопараллельной обработке, хотя с точки зрения конечного пользователя обеспечивается функциональная параллельность. Более детальное обсуждение этой проблемы было представлено в работе [3].

Для того чтобы подчеркнуть универсализм предлагаемого описания, рассмотрим несколько дополнительных примеров. Использование процессограммы для описания начальной фазы функционирования встроенной системы управления иллюстрирует рис. 3. На рис. 4 приведена процессограмма, описывающая диспетчеризацию с квантованием времени при использовании одного из системных таймеров. За время одного цикла опроса очереди задач длительностью T_c каждой из задач выделяется одинаковый квант времени T_k . Для реализации такого диспетчера mOCPB потребуется время на распознавание источника прерывания, время T_{isr} на его обработку, на сохранение контекста текущей выполняемой программы, на обновление счетчиков очередей самого диспетчера, на восстановление контекста следующей по очереди задачи. Все это займет время T_{disp}. Таким образом, полное время опроса очереди команд T_c с учетом «накладных расходов» определяется как

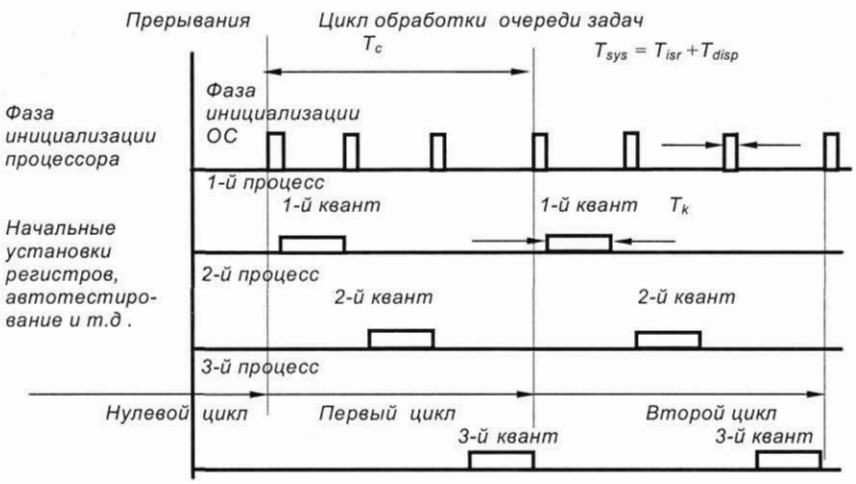
$$T_c = \sum_{i=1}^{N_{ch}} \left(T_k + (T_{isr} + T_{disp})\right) = \sum_{i=1}^{N_{ch}} (T_k + T_{sys}).$$

Для операционных систем с кооперативной организацией взаимодействия задач или систем с квантованием времени T_c представляет собой оценку времени реакции системы T_r на внешние воздействия по любому из параллельных каналов.

Системы диспетчеризации с приоритетами, как правило, широко используют систему прерываний. Возможность укрупненного описания алгоритма функционирования системы управления, использующей прерывания, с помощью процессограмм показана на рис. 5. Следует отметить, что, если речь идет о многоканальных системах управления «жесткого реального времени», диспетчеризация, использующая систему прерываний, приводит к целому ряду проблем, связанных с тем, что время ее реакции становится уже вероятностной характеристикой системы управления. Детальному рассмотрению круга проблем, связанных с использованием прерываний, посвящена работа [6].



■ ***Рис. 3.*** *Процессограмма начальной стадии работы многоканальной системы управления*



■ ***Рис. 4.*** *Процессограмма диспетчера с квантованием времени*

АТ-модель глобального конкурентного пространства

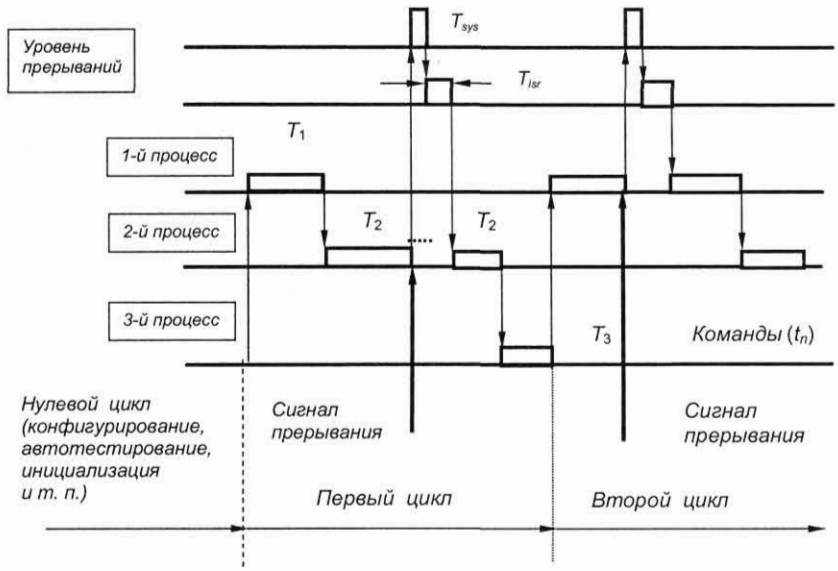
В современном стандарте на программное обеспечение POSIX понятию критической секции (critical section) соответствуют определения «конкурентное пространство» и «разделяемые ресурсы». В самом общем случае в конкурентное пространство входят: общие периферийные модули, используемые разными процессами, общие зоны памяти (вы-

зов одних и тех же подпрограмм из разных процессов), общие временные интервалы (для однопроцессорных систем). На рис. 6 представлен пример графического описания ресурсов современного микропроцессора, разделяемых параллельными процессами между собой в динамическом режиме. Для систем управления реального времени требуется описание временной последовательности реализации алгоритма за счет введения некоторой дискретизации и соответствующего описания оси времени, например, за счет номера выполняемого такта.

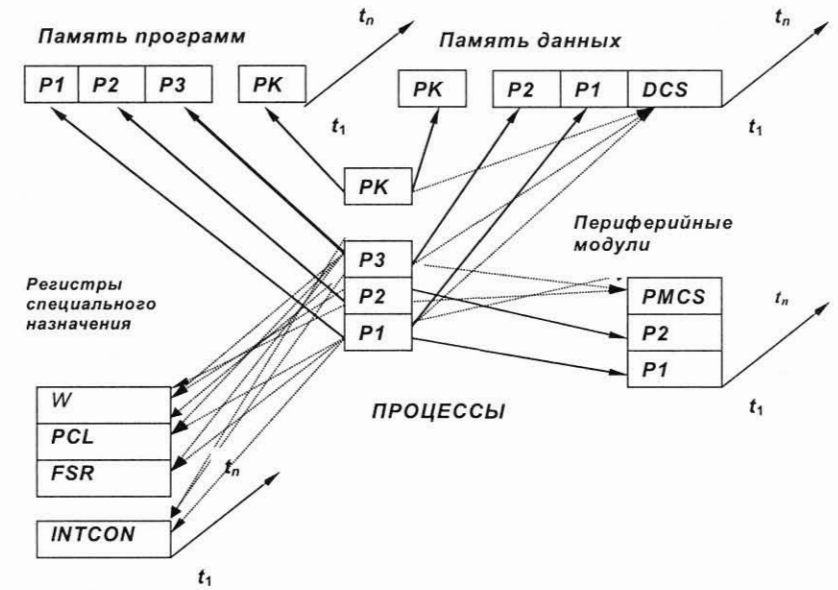
Описание разделяемых ресурсов как совокупности используемых адресов и временных тактов, задействованных конкретной задачей, процессом, каналом управления и т. п., будем называть адресно-временной моделью (AT-моделью от англ. Address-Time) глобального конкурентного пространства [3]. Такая AT-модель, представленная на рис. 6, описывает ресурсы микропроцессора с гарвардской архитектурой, в которой память программ и память данных разделены. Она же явно отражает факт динамически меняющейся ситуации, о чем свидетельствует ось времени, присовокупленная к каждому конкретному виду разделяемых между процессами ресурсов. По оси времени отображаются дискретные отсчеты, что соответствует тактам микропроцессора, реализующего алгоритм управления.

На низком уровне описание алгоритмов функционирования осуществляется на ассемблере, в котором каждому из регистров присваивается уникальное символьное имя. Для упорядочения всех доступных ресурсов микропроцессора представляется разумным присваивать уникальные номера всем регистрам, а также элементам ядра микропроцессора, которые, по сложившейся к настоящему времени традиции, будем называть адресами. Это позволяет однозначно идентифицировать каждый элемент микропроцессора.

Проблема синтеза алгоритмов функционирования многоканальных систем управления реального времени требует разработки



■ ***Рис. 5.*** *Процессограмма описания диспетчеризации с приоритетами*



■ ***Рис. 6.*** *Структура глобального конкурентного пространства:*

P1–PK — процесс № 1–K; DCS — общее пространство памяти данных (Data Concurrent Space); PMCS — разделяемые периферийные модули (Peripheral Module Concurrent Space); t_1–t_n — временные отсчеты, согласованные с тактовой частотой процессора; W, PCL, FSR, INTCON — регистры: рабочий, счетчика команд, неявной адресации и контроля прерываний, соответственно

эффективных методов контроля за разделяемыми ресурсами. Представляется целесообразным разделить ее на следующие задачи: разработка логической структуры алгоритма по каждому из каналов; бесконфликтное разделение адресно-временных ресурсов между параллельно выполняющимися задачами.

При наличии библиотек прикладных программ для выполнения типовых задач систем управления стоит проблема корректного переноса соответствующих фрагментов кода в тело разрабатываемого программного обеспечения. Это, в свою очередь, требует корректного разделения адресного пространства между микрооперации-

онной системой, прикладными задачами, реализующими логическую структуру алгоритма управления, и задачами управления периферийными устройствами.

Отдельную проблему представляет задача обеспечения мобильности программного обеспечения в связи с необходимостью переопределения адресов при переносе фрагментов кода на различные устройства с однотипными функциями, но отличающиеся друг от друга по схемотехнике.

В этом случае в зависимости от конкретных условий может потребоваться решение не только задачи бесконфликтного разделения адресного конкурентного пространства, но и задачи корректного разделения адресно-временного конкурентного пространства. Можно также говорить о постановке оптимизационной задачи по выбору типа используемого микропроцессора.

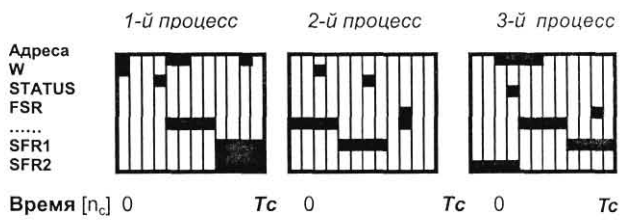
Адресно-временная карта процесса

АТ-модель глобального конкурентного пространства допускает простой способ формального описания. Единообразное и универсальное описание глобального конкурентного пространства может быть получено за счет присвоения каждому элементу конкурентного пространства своего номера и дискретизации временной оси в соответствии с монотонно возрастающим числом командных циклов, начиная с некоторого начального отсчета. В двумерном адресно-временном множестве, элементы которого распределяются между параллельно выполняющимися процессами, каждому из них будет соответствовать некоторое подмножество, которое в дальнейшем будем называть адресно-временной картой процесса или АТ-картой.

Возможность такого представления для описания распределения используемых регистров между параллельно выполняющимися процессами совершенно естественна для микропроцессоров, использующих концепцию регистрового файла, и требует некоторых предварительных соглашений о нумерации регистров для других архитектур. Поскольку любой процесс состоит из задач, каждая из которых представляет собой некоторую совокупность последовательно выполняемых команд, то описание временной оси можно представить в дискретном виде. Соответствующая АТ-карта k-го процесса представляется в виде прямоугольной матрицы описания AT_k с элементами:

$$AT_k(i, j) = \begin{cases} 1 — \text{если } k\text{-й процесс в } j\text{-м такте} \\ \quad \text{использует } i\text{-й адрес} \\ \quad \text{микропроцессора;} \\ 0 — \text{в противном случае.} \end{cases}$$

На рис. 7 приведена графическая иллюстрация предлагаемого способа описания используемых ресурсов микропроцессора с помощью АТ-карт. Ее можно применять и для описания алгоритмов функционирования распределенных (сетевых) систем управления. При этом АТ-карты описания ресурсов,

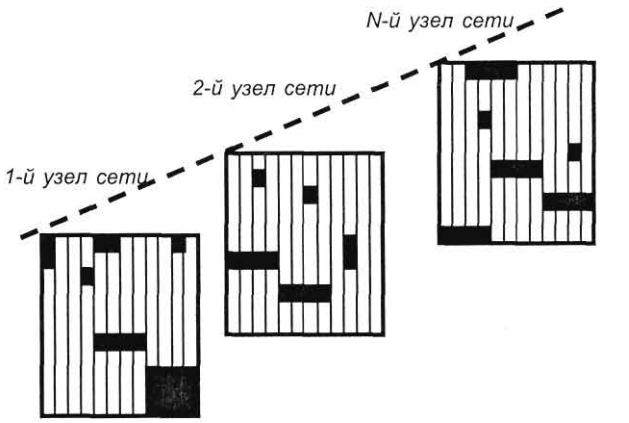


■ ***Рис. 7.*** *Иллюстрация к определению понятия «адресно-временная карта»*

используемых каждым из узлов, образуют трехмерную АТ-карту, как это показано на рис. 8. В этом случае логично ввести понятие AT_N-карты, представляющей собой трехмерную матрицу описания системы параллельно выполняющихся процессов. Символ N соответствует порядковому номеру узла сети.

Предложенный способ описания параллельных процессов с помощью адресно-временных карт и матриц позволяет достаточно просто сформулировать критерий наличия конфликтной ситуации, который к тому же достаточно просто алгоритмизируется. В первом приближении наличие конфликта между параллельными процессами для корректно сгенерированной прикладной программы означает использование одного и того же адреса в один и тот же квант времени более чем одним процессом [3]. Поставим каждому из параллельно выполняющихся процессов в соответствие АТ-карту, описывающую использование конкретных адресов микропроцессора в конкретных командных циклах, отсчитываемых от предписанного начального значения (например, от начала опроса очереди задач). Тогда при суммировании матриц описания карт адресно-временных подмножеств, соответствующих разным процессам, получим формальный и алгоритмически простой критерий отсутствия конфликтов между параллельно выполняющимися процессами в одном цикле опроса очереди задач в виде

$$\max_{i,\,j} \sum (AT_k) = 1.$$



■ ***Рис. 8.*** *Вариант использования трехмерных АТ-карт (AT_N-матриц) для описания алгоритмов функционирования сетевых систем управления*

В качестве меры в пространстве конфликтов можно использовать значение максимального элемента суммы АТ-карт процессов.

При использовании алгоритмов обработки разреженных матриц, хорошо разработанных для численного решения многомерных задач математической физики, матрица описания АТ-карты представляется в виде трехмерного массива (или трех одномерных массивов равной размерности) описания ненулевых элементов в виде

$$AT = \{i, j, V_{i,j}\}. \qquad (2)$$

Фигурные скобки в выражении (2) означают массивы описания АТ-карты, организованные тем или иным способом: в виде трехмерного массива, в виде трех одномерных массивов, в виде структуры и т.п. Для современных микропроцессоров значение размерности адресной компоненты лежит в интервале от 100 до 10 000 0000, а количество командных циклов определяется спецификой решаемой задачи. Представление (2) можно усовершенствовать за счет более рационального способа описания АТ-карты. Если трактовать $V_{i,j}$ как количество ненулевых элементов карты, начиная с элемента (i, j), то удастся существенно сократить общий объем описаний требуемых ресурсов. Согласно этому способу, наличие тройки $(i, j, V_{i,j})$ в описании АТ-карты означает, что адрес i занят на протяжении $V_{i,j}$ тактов, начиная с j-го. Так как описываются только ненулевые компоненты матрицы, то для полноты описания потребуется знание размерности массивов описания (например, N_1 для AT_1), а также границ A_{beg}, A_{end}, T_{beg} и T_{end}. При таком способе представления ненулевых элементов описание АТ-карты представляет собой структуру, содержащую размерности $[N_A, N_T, N_1]$ и трехмерный массив описания индексов ненулевых элементов $\{i, j\}$:

$$AT_k = [N_A, N_T, N_1]\ \{i, j, V_{i,j}\}_k, \qquad (3)$$

где квадратные скобки означают структуру, содержащую размерности описания k-й АТ-карты. Речь идет о структуре в смысле определения элементов данных, используемых в языке программирования С.

Поскольку элементы $V_{i,j}$ матриц полного описания АТ-карт могут принимать лишь два значения (0 или 1), то описание (2) без потери информативности может быть представлено индексами i и j. При этом предполагается, что наличие такой пары означает, что соответствующий элемент матрицы описания $V_{i,j}$ равен 1. При таком способе представления ненулевых элементов описание АТ-карты представляет собой тройку размерностей (N_A, N_T, N_1) и двумерный массив описания индексов ненулевых элементов $\{i, j\}$ в виде

$$AT_k = (N_A, N_T, N_1)\ \{i, j\}_k. \qquad (4)$$

При этом предполагается, что наличие пары (i, j) в описании означает, что соответствующий элемент АТ-карты равен 1, хотя в принципе не исключено и применение инверсного способа описания используемых адресов. Понятно, что рассмотренные выше представления одномерны и относятся к временной оси АТ-карты. Аналогичные способы описания могут быть использованы для адресной компоненты АТ-карты. В этом случае дальнейшие обобщения могут потребоваться при описании некоторой зоны памяти на протяжении заданного числа циклов, что приведет к следующей базовой структуре описания в виде:

$$AT_k = \{i_{beg}, j_{beg}, A_{i,j}, T_{i,j}\}_k, \qquad (5)$$

которая подразумевает, что адреса процессора с i_{beg} по $i_{beg} + A_{i,j}$ на моменты времени с j_{beg} по $j_{beg} + T_{i,j}$ закреплены за k-й задачей.

Универсальный и развиваемый способ описания АТ-карт представляет собой возможность использования выражений (2) – (5) и ряда других, потребность в которых может возникнуть при решении конкретных задач. В этом случае для каждого типа описаний потребуется введение своего идентификатора, а выполнение операций с АТ-картами будет представлять собой операции с массивами описаний, определяемыми непосредственно типом выполняемой операции и способом описания конкретной АТ-карты. Трудоемкость решения прикладных задач в смысле количества требуемых операций будет зависеть от специфики решаемой задачи, но с точки зрения прикладной математики это сводится к хорошо изученным задачам сортировки массивов, а также переборным алгоритмам.

Символьный способ описания

При более глубоком рассмотрении проблемы анализа реализуемости алгоритмов с точки зрения обеспечения специфицированных времен реакции на внешние воздействия оказывается, что одного лишь численного представления недостаточно. В качестве иллюстрации этого рассмотрим одну важную проблему, с которой приходится сталкиваться на практике. Зачастую встроенные системы управления работают в необслуживаемом режиме на протяжении длительного времени.

В качестве простейшего примера можно привести, например, многотарифный трехфазный счетчик электроэнергии, который должен обеспечивать измерение значения потребляемого тока, рассчитывать мгновенную потребляемую мощность по каждой фазе и количество потребленной энергии по каждой тарифной зоне, обеспечивать коммуникационные процедуры служб технической поддержки, обрабатывать сбои и т. п. Общее число параллельно обрабатывающихся процессов в такой системе составляет от 20 до 30. Важно отметить, что межповерочный интервал такого класса устройств составляет 10 лет, при этом, как правило, должна быть обеспечена точность измерений порядка 0,5–1,0 %.

Микропроцессор такой системы конфигурируется на самой первой стадии работы, далее ини-

циализируется mOCPB и контролируемые ею задачи. От корректных значений параметров инициализации задач зависит корректность работы всей системы управления и устройства в целом. Для корректной работы диспетчера, который является частью mOCPB, если позволяют временные характеристики, следует включить в цикл опроса очереди задач и задачу инициализации процессора, для обеспечения гарантированных начальных условий выполнения задач, образующих процессы. В целях экономии циклов процессора на инициализацию возможна диспетчеризация посредством двух вложенных очередей, что иллюстрирует пример 1. Здесь использован синтаксис ассемблера микропроцессорной платформы PIC16/17.

Пример 1. Диспетчер для двух вложенных очередей

```
        bcf     INTCON,GIE      ; запрет прерываний
LINE0   call    INIT            ; инициализация процессора,
                                ; диспетчера и задач
LINE1   call    TASK1           ; вызов 1 задачи
        call    TASK2           ; вызов 2 задачи
        .....
        call    TASKN           ; вызов задачи N
        call    COUNT           ; вызов служебной
                                ; подпрограммы диспетчера
                                ; подсчета числа проходов
        btfsc   PSW, INIT_FLAG  ; бит флага необходимости
                                ; инициализации
goto    LINE0                   ; устанавливается
                                ; подпрограммой COUNT
goto    LINE1                   ; формирование временного
                                ; процесса
```

Диспетчеризация осуществляется таким образом, что во внутреннем цикле, обрамленном меткой LINE1 и оператором goto LINE1, осуществляется поочередный вызов задач. С помощью служебной программы COUNT, являющейся принадлежностью диспетчера и соответственно mOCPB, производится подсчет числа проходов по внутреннему циклу. При равенстве числа проходов некоторому заданному значению подпрограмма COUNT выставляет соответствующий бит INIT_FLAG в регистре PSW. В конце каждого прохода очереди команд производится проверка этого флага. Диспетчер, обнаружив, что флаг выставлен, передает управление во внешний цикл, в котором происходит реинициализация процессора, диспетчера, задач. В этом модельном примере рассматривается диспетчеризация без прерываний, и для этого перед началом работы система прерываний блокируется выставлением бита GIE в регистре управления системой прерываний INTCON.

Расширение приведенного подхода возможно, во-первых, на большее число задач в редко выполняющейся очереди, начинающейся с метки LINE0, во-вторых, на большее число вложенных очередей. Каждая вложенная очередь потребует отдельного регистра (по крайней мере, одного) на организацию счетчика, а также от 4 до 10 операций за один цикл выполнения очереди задач, которые уйдут на его инкрементирование, проверку его переполнения и передачу управления. Следует обратить внимание на то, что использование многопетлевого диспетчера в приведенном примере с включением во внешние циклы объемных по времени задач, существенным образом ухудшает время реакции системы управления T_r. В случае двухпетлевого диспетчера в качестве оценки времени реакции системы управления будет фигурировать время, требуемое на инициализацию задач, суммированное со временем их выполнения.

Приведенный пример предоставляет возможность рассмотреть еще один аспект использования аппарата АТ-карт для описания разнообразных дисциплин диспетчеризации. Если между частотой вызовов подпрограмм внутреннего цикла и инициализацией имеется существенное различие, то процессограммы и соответственно АТ-карты, отображаемые непосредственно на мониторе, много теряют в своей наглядности. Для того чтобы добраться до цикла инициализации, придется достаточно долго «листать» экраны. С точки зрения потребностей реальной практики, целесообразно развивать способы символического описания суммарных АТ-карт. Идею их использования иллюстрирует формула (6), описывающая циклограмму диспетчера из примера 1:

$$AT_S = \sum_{k=1}^{\infty} AT_0 + \sum_{\mathrm{COUNT}=1}^{N_1} \sum_{i=1}^{N_t} AT_i . \qquad (6)$$

Один цикл прохода очереди задач описывается суммарной картой, сформированной из карт AT_i, общее число которых составляет N_t. Эти карты соответствующим образом складываются N_1 раз, образуя суммарную карту внутреннего цикла. К этой карте добавляется карта AT_0, описывающая процесс инициализации. Сформированная АТ-карта репродуцируется бесконечное количество раз, что отражает символ ∞ во внешнем цикле суммирования. При детальном рассмотрении оказывается, что использование напрямую операций суммирования матриц в соответствии с классической теорией матриц возможно, но неудобно. Поэтому необходимо разработать символьный способ описания АТ-карт, а также ввести иерархию операций для работы с прямоугольными матрицами, что детально будет рассмотрено далее [3].

Представление (6) позволяет компактно и обозримо представить алгоритм формирования суммарной АТ-матрицы, обеспечивая тем самым доступ к любому фрагменту циклограммы. Необходимо также отметить, что альтернативы этому способу нет, так как даже при самом экономном способе организации хранения элементов АТ-карт невозможно будет хранить АТ-карту бесконечного цикла. В соответствии с изложенным становится ясно, что при разработке соответствующего программного инструментария для работы с АТ-картами целесообразно ориентироваться на применение комбинированных представлений в виде иерархических структур из численных и символьных описаний.

Классификация конфликтов в глобальном конкурентном пространстве

Из-за сложного характера информационного взаимодействия детальное описание алгоритма функционирования систем управления с числом каналов управления, превышающим 10, становится громоздким. Это обусловлено необходимостью рассматривать сложное поведение системы при широком спектре входных воздействий на длительных интервалах времени. Рассмотрим возможности использования АТ-карт для анализа конфликтных ситуаций. Начнем с более детального рассмотрения определения конфликта в глобальном конкурентном пространстве, введя его в следующем виде.

Определение 1. Наличие АТ-конфликта между параллельными процессами для корректно сгенерированной прикладной программы означает попытку использовать один и тот же адрес в один и тот же командный цикл более чем одним процессом. При суммировании матриц описания карт адресно-временных подмножеств, соответствующих разным процессам, получаем легко формализуемый и алгоритмически простой критерий отсутствия конфликтов между параллельно выполняющимися процессами в виде

$$\max_{i,\,j} \sum_{k} (AT_k) = 1, \qquad (7)$$

где i, j — индексы для описания элементов АТ-карт (i — индекс описания адреса; j — индекс нумерации столбцов в командных циклах); k — индекс в описании количества параллельно выполняющихся процессов.

Использованная в выражении (7) операция суммирования матриц представляет собой обыкновенную операцию поэлементного суммирования матриц и подразумевает, что суммируемые матрицы имеют одинаковые размерности. Это накладывает следующие ограничения на выбор формы описания АТ-карт. Во-первых, следует использовать полное адресное пространство микропроцессора в совокупности с регистрами специального назначения. Во-вторых, в качестве начального отсчета, соответствующего первому столбцу каждой из АТ-карт, следует брать командный цикл, соответствующий запуску микрооперационной системы. В-третьих, требуется выбор отрезка временной оси, на котором будет проводиться анализ на наличие конфликта, так как для большинства типовых применений суммарная матрица описания АТ-карты бесконечна.

Отметим, что выполнение условия (7) соответствует отлаженной программе. При автоматизированном синтезе ППО для многоканальной системы управления потребуется решение задачи разделения ресурсов между параллельными процессами, так как с большой вероятностью на первых этапах синтеза АТ-карты разных процессов будут иметь пересечения. В этом случае максимум $F(*,*) = \max \sum (AT_k)$ можно использовать в качестве функционала, численно отражающего конфликтность ситуации. В качестве аргументов в этом функционале будут выступать свободные для подбора параметры, определяющиеся конкретной задачей. Использование такого функционала обеспечивает формальное сведение задачи бесконфликтного разделения адресно-временных ресурсов к классической задаче целочисленного программирования. Однако интуитивно ясно, что функционал явно не будет попадать в класс унимодальных, и на практике в дальнейшем потребуется разработка соответствующих специализированных численных методов.

Критерий наличия конфликта в адресном пространстве для задач, описываемых картами AT_1 и AT_2, может быть сформулирован с использованием аппарата классической теории множеств в следующем виде.

Определение 2. Наличие А-конфликта между двумя программами означает, что пересечение множества адресов, используемых каждой из подпрограмм, не пусто, т. е. $\{i\}_1 \cap \{i\}_2 \neq 0$.

Вводя соответствующий формализм, определим множества $\{i\}$ как проекции АТ-карт на адресную ось, что может быть записано в виде

$$[\perp A]\, AT_1 = \{i\}_1,\ [\perp A]\, AT_2 = \{i\}_2. \qquad (8)$$

Квадратные скобки в выражении (8) использованы для удобства обозначения и восприятия конкретной операции. А-конфликт служит характеристикой того, что разные процессы имеют попеременный доступ к одним и тем же ресурсам, например, переменным. В рамках использования предлагаемого формализма это означает совпадение индексов i в описаниях АТ-карт, принадлежащих разным задачам. Понятно, что в силу самой постановки задачи предлагаемое описание не учитывает ни семантику вычислений, ни алгоритмические аспекты вычислений. Вместе с тем введение такого рода конфликта позволяет автоматизировать поиск ошибочно введенных глобальных переменных, которые являются неиссякаемым источником ошибок при интеграции программистских проектов.

При этом речь идет не обязательно о системах управления реального времени. Использование адресной составляющей АТ-карт позволяет провести анализ на совпадение символьных имен (адресов) в рамках интеграции отработанных процедур в состав вновь разрабатываемого ППО. При этом могут решаться задачи бесконфликтного разделения как памяти, так и периферийных устройств: портов ввода-вывода, коммуникационных модулей, модулей АЦП и т. п. Дальнейшее движение по этому пути позволяет сформулировать не только задачу бесконфликтного разделения ресурсов при заданном типе микропроцессора, но и задачу выбора микропроцессора.

По аналогии с А-конфликтом может быть введена операция проекции АТ-карты на ось временных отсчетов в виде $[\perp T]\, AT_1 = \{j\}_1$.

Определение 3. Наличие Т-конфликта между двумя программами означает, что пересечение мно-

жества временных отсчетов, используемых каждой из программ, не пусто. В этом случае критерий наличия конфликта между двумя процессами в пространстве временных отсчетов записывается в виде $\{j\}_1 \cap \{j\}_2 \neq 0$.

Такого рода характеристика может потребоваться при проектировании коммуникационных процедур для многоканальных систем управления. В этом случае наличие Т-конфликта означает потерю информации по одному из конфликтующих каналов. Понятие Т-конфлита позволяет формулировать как задачу проверки адекватности выбора тактовых частот, так и соответствующие задачи оптимизации.

Введенная система определений обеспечивает следующие классификации конфликтов в глобальном конкурентном пространстве: Т- конфликт, А-конфликт, АТ- конфликт. Очевидно, что одновременное отсутствие А-конфликта и Т-конфликта является достаточным условием корректности программы при использовании критической секции. Соответственно явно просматриваются способы построения численных процедур автоматического синтеза программного обеспечения для многоканальных систем управления реального времени. Развитие соответствующих методов и методик автоматизированного решения подобного класса задач следует вести параллельно с разработкой соответствующих программно-инструментальных средств, что представляет собой задачу следующего шага.

Литература

1. **Астапкович А. М.** Аналитический обзор: высокоуровневые программно-инструментальные среды для встраиваемых приложений реального времени // Микропроцессорные информационно-управляющие системы реального времени. Семинары ASK Lab 2000/Под ред. М. Б. Сергеева. — СПб.: Политехника, 2000. — С. 4–39.
2. **Астапкович А. М., Востриков А. А., Гуляев А. М.** Аналитический обзор: операционные системы реального времени для встраиваемых приложений // BYTE Россия, 2000. — № 9. — С. 29–37.
3. **Астапкович А. М.** Микрооперационные системы (mOCPB) для встроенных систем управления // Информационно-управляющие системы и сети. Структуры. Моделирование. Алгоритмы / Под ред. М. Б. Сергеева. — СПб.: Политехника, 1999. — С. 196–224.
4. **Основы** имитационного и статистического моделирования / Ю. С. Харин и др. — Минск: Дизайн ПРО, 1997. — 288 с.
5. **Игнатьев М. Б., Фильчаков В. В., Осовецкий Л. Г.** Активные методы обеспечения надежности алгоритмов и программ. — СПб.: Политехника, 1992. — 285 с.
6. **Астапкович А. М.** Вероятностное поведение многоканальных систем управления реального времени и логическое проектирование алгоритмов//Микропроцессорные информационно-управляющие системы реального времени. Семинары ASK Lab 2000/ Под ред. М. Б. Сергеева. — СПб.: Политехника, 2000. — С. 104–134.