УДК 004.94

АНАЛИЗ МОДЕЛИ СИСТЕМЫ РЕАЛЬНОГО ВРЕМЕНИ НА СПОСОБНОСТЬ ВЫПОЛНИТЬ ПРИКЛАДНЫЕ ФУНКЦИИ ПРИ ЗАДАННЫХ УСЛОВИЯХ ДИНАМИКИ РАБОТЫ ОБЪЕКТА УПРАВЛЕНИЯ

В.К. Погребной, А.В. Погребной, Д.В. Погребной*, В.А. Дорофеев

Институт «Кибернетический центр» ТПУ
*ООО «Контек-Софт», г. Томск
E-mail: Pogrebnoy@tpu.ru

Предложены аналитические методы анализа временных характеристик работы модели системы реального времени, представленной на языке структурного моделирования в форме графа потока данных. Исследованы возможности автономного выполнения процессов и прикладных функций системы за время, которое не противоречит заданным условиям динамики функционирования объекта управления. Анализ совместной работы прикладных функций и соответствующих процессов выполнен в условиях распараллеливания модели и конвейерной схемы применения процессоров. Результаты анализа позволяют разработчику проекта оценить способность проектируемой системы своевременно выполнять прикладные функции.

Ключевые слова:
Модель системы реального времени, граф потока данных, ярусно-параллельная форма, процесс, прикладная функция, конвейерная схема, вычислительная система.

Key words:
Realtime system model, dataflow graph, multilevel structure, process, applied function, pipeline scheme, computer system.

При проектировании распределенных систем реального времени (СРВ) разработчик проекта вынужден использовать эволюционный подход, когда исходная модель системы последовательно трансформируется так, чтобы ее конечный вариант соответствовал системе с приемлемыми характеристиками. В [1] представление модели СРВ предложено осуществлять на языке структурного моделирования (языке SML), а математические и программные средства построения и эволюции модели названы SML-технологией. Существенным для SML-технологии является то, что в ней большая часть трансформаций осуществляется аналитическими методами, а имитационное моделирование используется на заключительной стадии эволюции модели. Решению одной из задач аналитического моделирования на начальном этапе эволюции модели СРВ посвящена данная работа. Задача заключается в том, чтобы для представленного варианта модели оценить способность соответствующей СРВ выполнить заданную совокупность прикладных функций (ПФ) по управлению объектом с соблюдением условий динамики его функционирования.

Модель программной нагрузки СРВ, реализующей заданную совокупность ПФ с учетом их совместной работы, построена с помощью визуального уровня языка SML в форме графа потока данных (ГПД). Модель исходного варианта архитектуры вычислительной системы, на которой должен выполняться ГПД, представлена совокупностью микропроцессорных станций (контроллеров), подключенных к одной магистрали. Число станций в исходном варианте вычислительной системы определяется как минимально возможное исходя из необходимости подключения всех терминальных точек объекта управления [2].

Методика решения задачи по оценке способности СРВ своевременно выполнять ГПД на вычислительной системе включает три последовательно выполняемых этапа. Каждый последующий этап, в сравнении с предыдущим, более полно учитывает условия динамики функционирования СРВ. Параметры динамики диктуются объектом управления и перед выполнением анализа переносятся на модель СРВ [3]. Процесс переноса не является однозначным и предполагает наличие разных вариантов построения временных диаграмм, как при автономном, так и совместном выполнении ПФ. В последующем этапы анализа будут рассмотрены относительно одного варианта значений параметров динамики модели СРВ.

1. Анализ автономного выполнения процессов

Модель СРВ, учитывающая параметры динамики, представляется совокупностью динамических объектов (процессов), которые могут функционировать параллельно и взаимодействовать между собой. Этапы анализа будем рассматривать на примере ГПД (рис. 1), содержащего 28 позиций (данных) и 17 переходов (модулей) объединённых в 9 процессов. На рис. 1 процессы выделены пунктирными линиями.

Представим ГПД двудольным графом $R=(F, D, S)$, где $F=\{f_m\}$, $m=1,2,...,M$ – множество модулей; $D=\{d_q\}$, $q=1,2,...,Q$ – множество данных; S – множество дуг (d_q,f_m) или (f_m, d_q), связывающих модули и данные.

Входные данные, в зависимости от условий поступления, разбиваются на 4 группы: *Ц* – циклическое поступление, *В* – вероятностное поступление, *Д* – детерминированная последовательность моментов поступления, *У* – поступление по заданному условию. Для каждого входа d_q, в зависимости от

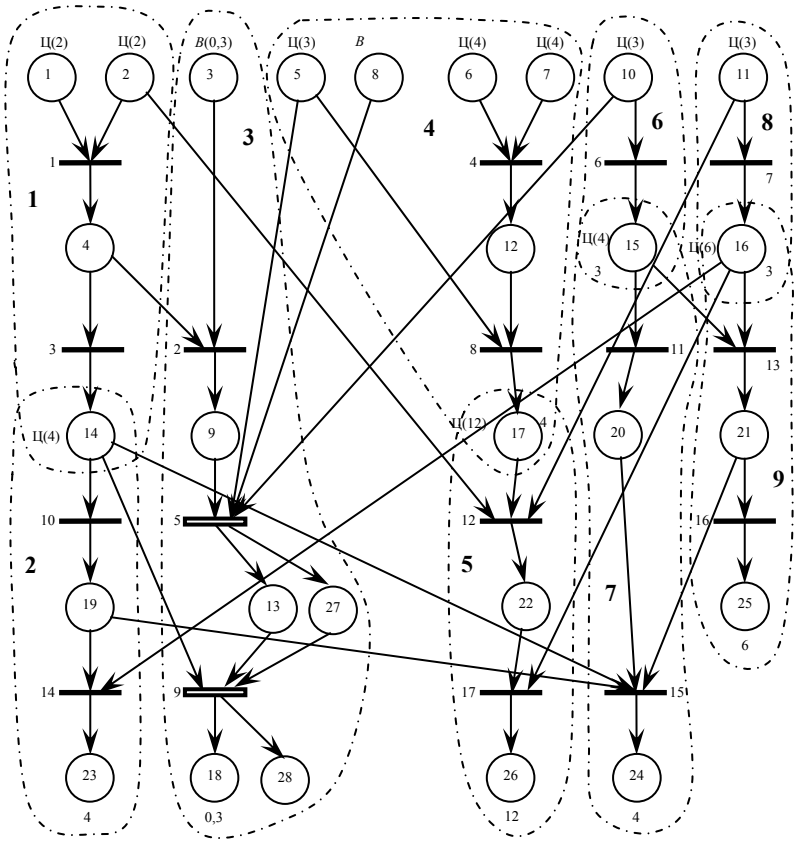

Рис. 1. *Пример ГПД в ярусно-параллельной форме*

принадлежности к одной из групп, указывается соответствующий интервал времени, $\delta_{Ц}(d_q)$, $\delta_{В}(d_q)$, $\delta_{Д}(d_q)$, $\delta_{У}(d_q)$. Выходные данные d_q сопровождаются указанием одного из интервалов времени $\mu_{Ц}(d_q)$, $\mu_{В}(d_q)$, $\mu_{Д}(d_q)$, $\mu_{У}(d_q)$.

Число станций ВС, необходимых для выполнения данного ГПД, принято равным 6. Распределение модулей и данных по станциям приведено в таблице. Здесь же для каждого модуля f_m в скобках указано время его выполнения τ_m.

Таблица. *Распределение модулей и данных по станциям*

Станция	Модули f_m	Данные d_q
1	1(0,5), 2(1), 3(1)	1, 2, 3, 4, 9
2	4(1,5), 8(2)	5, 6, 7, 12, 17
3	5(0,5), 9(1,5), 10(2), 14(1)	8, 13, 14, 18, 19, 23, 27, 28
4	12(5), 17(6)	22, 26
5	6(0,5), 7(0,5), 11(2)	10, 11, 15, 16
6	13(2), 15(1), 16(2)	20, 21, 24, 25

Основная цель данного этапа анализа заключается в проверке возможности автономного выполнения процесса в установленное время. Для каждого процесса p известна величина $\mu_{Ц}(p)$, если он запускается по правилу *Ц* или величины $\mu_{В}(p)$, $\mu_{Д}(p)$, $\mu_{У}(p)$, если процесс p запускается по одному из правил *В*, *Д*, *У* соответственно. Ограничение на время выполнения процесса p независимо от правила его запуска обозначим величиной $\tau(p)$. Время выполнения процесса p зависит от совокупности F_p, входящих в него модулей, и необходимости передачи данных по сети, если для выполнения модулей привлекается более одной станции. В общем случае процедура выделения процессов разбивает граф R на подграфы $R_p=(F_p,D_p,S_p)$, каждый из которых соответствует процессу p с множеством данных D_p и модулей F_p. Заметим, что для любых процессов p_1 и p_2, $F_{p_1}\cap F_{p_2}=\varnothing$, а $D_{p_1}\cap D_{p_2}(\neq,=)\varnothing$ то есть пересечение множеств данных может быть и не пустым. Например, из рис. 1 следует, что $D_4\cap D_5=d_{17}$.

В графе R_p сформируем все возможные маршруты выполнения процесса с началом в вершине запуска и концом в выходной вершине, имеющей ограничение на время получения $\tau(p)$. Время автономного выполнения процесса p оценивается по каждому маршруту и сравнивается с величиной $\tau(p)$. Если совокупность модулей k-го маршрута F_{pk} выполняется на разных станциях, то такой маршрут содержит совокупность $S_{pk}\neq\varnothing$ дуг (d_q, f_m), по которым данные в объемах r_{qm} должны передаваться по сети, затрачивая время τ_{qm}. Таким образом, для каждого k-го маршрута процесса p должно выполняться условие:

$$\sum_{f_m\in F_{pk}} \tau_m + \sum_{(d_q, f_m)\in S_{pk}} \tau_{qm} \le \tau_p . \qquad (1)$$

Рассмотрим применение условия (1) на примере ГПД (рис. 1) для анализа 7-го процесса. Граф R_7 процесса показанный на рис. 2, *а*, содержит один маршрут с модулями f_{11} и f_{15}, которые выполняются

на станциях 5 и 6 соответственно, $F_{pk}=(f_{11},f_{15})$, $S_{pk}=(f_{20},f_{11})$.

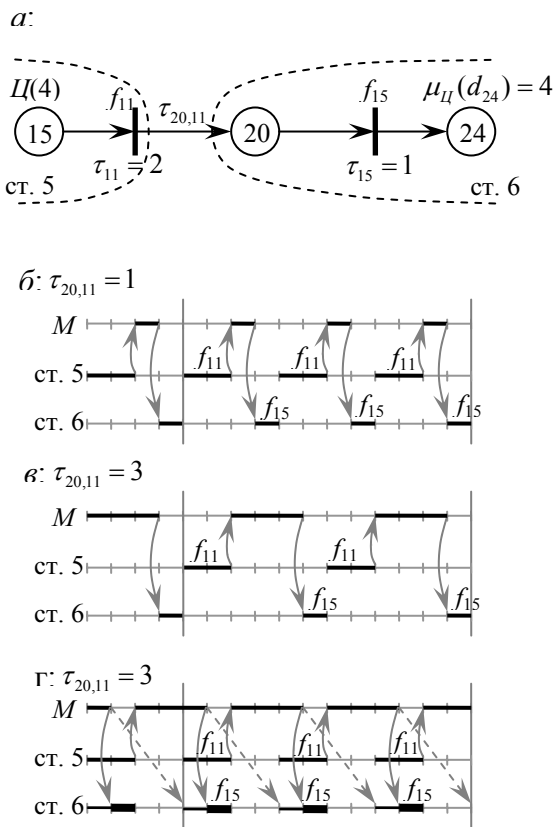

Рис. 2. *Пример анализа процесса*

На рис. 2, *б*, приведена диаграмма автономного выполнения 7-го процесса для случая, когда время передачи данных $\tau_{20,11}=1$, а цикл моделирования $T=12$ тактов. При выполнении процесса соблюдается правило запуска модуля f_{15} по наличию данных (в примере – это d_{20}). Из диаграммы следует, что время выполнения процесса не превышает заданное ограничение $\mu_Ц(d_{24})=\tau(7)=4$. На диаграмме (рис. 2, *в*) показано, что для $\tau_{20,11}=3$ такта правило запуска по наличию данных неприемлемо, так как время с момента запуска процесса (модуль f_{11}) до получения выхода d_{24} составляет 6 тактов и, следовательно, за один цикл моделирования процесс выполняется 2 раза, а не 3, как это задано величиной $\mu_Ц(d_{24})=4$. Обе диаграммы отражают последовательное выполнение модулей f_{11} и f_{15} и могли быть получены при выполнении процесса на одной станции.

Преимущество, которое можно получить при организации выполнения процесса на двух станциях, показано на диаграмме (рис. 2, *г*). Диаграмма отражает использование конвейерной схемы при выполнении модулей процесса. Здесь модули запускаются параллельно через 4 такта и используют данные, полученные на момент запуска. На рис. 2, *г,* пунктирными линиями показано, что модуль f_{15} в момент запуска имеет возможность использовать состояние позиции d_{20}, полученное модулем f_{11} на 2 цикла ранее. В этом случае время выполнения процесса увеличивается до 9 тактов. Ситуацию можно улучшить, если модуль f_{15} после запуска в течение одного такта будет ожидать завершения передачи данного d_{20}, полученного в предыдущем цикле. Время выполнения процесса при этом составит 6 тактов, что противоречит условию (1), но в цикле моделирования процесс выполняется 3 раза, т. е. условие обновления $\mu_Ц(d_{24})=4$ соблюдается.

Если условие (1) для какого-либо процесса не выполняется, то по отношению к нему применяются действия по сокращению времени выполнения. Среди них написание более эффективных по быстродействию программ, реализующих модули процесса, упрощение алгоритмов модулей и, наконец, разбиение модулей на фрагменты и их параллельное выполнение. Не исключается также вариант увеличения числа станций.

2. Анализ параллельного выполнения ГПД

Более полная проверка соблюдения ограничений по циклам обновления выходных данных производится в условиях параллельной работы ГПД. С этой целью ГПД преобразуется в ярусно-параллельную форму [4], рис. 1. При построении ярусно-параллельной формы ГПД рассматривается как модульная структура без учета условий поступления входных данных и обновления выходных, а также без выделения процессов, согласующих эти параметры. Методика построения заключается в последовательном формировании ярусов, содержащих модули способные выполняться на разных станциях параллельно. Первый ярус образует множество всех тех модулей, которые используют только внешние входные данные, обозначим их множеством X^0. Второй ярус включает модули, которые используют входные данные из множества X^0 и выходные данные модулей первого яруса. Аналогично любой последующий ярус i формируется из модулей, использующих входные данные из множества X^0, и выходные данные модулей, расположенных на любом из сформированных ранее ярусов в том числе и $(i-1)$-м ярусе. Для рассматриваемого примера ГПД модули по i-м ярусам с множествами F_i распределились следующим образом: $F_1=(f_1, f_4, f_6, f_7)$, $F_2=(f_2, f_3, f_8, f_{11}, f_{13})$, $F_3=(f_5, f_{10}, f_{12}, f_{16})$, $F_2=(f_9, f_{14}, f_{15}, f_{17})$.

На рис. 3 показана временная диаграмма одного цикла работы ГПД в соответствии с ярусно-параллельной формой представления. Диаграмма построена исходя из запуска модулей при наличии соответствующих данных. Использовано 6 станций, подключенных к одной магистрали. Распределение модулей и данных ГПД по станциям принято согласно таблице. На диаграмме показаны модули, которые запускаются циклически. Модули f_2,f_5,f_9 запускаются ациклически и для них ресурсы вычислительной системы резервируются отдельно.

При построении диаграммы время на пересылку для всех данных принято одинаковым и рав-

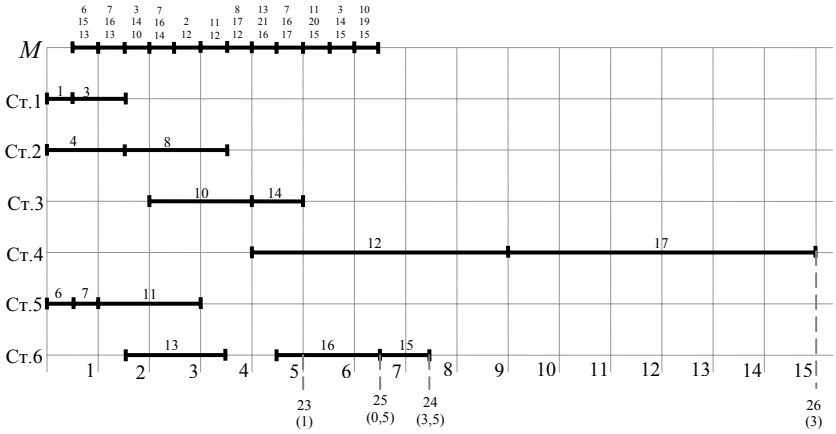

Рис. 3. *Временная диаграмма, построенная для одного цикла работы ГПД*

ным 0,5 такта. Надписи над временной осью магистрали указывают, какие данные и для каких модулей передаются от одной станции к другой. Например, запись 6–15–13 соответствует передаче данных d_{15}, полученных модулем f_6, для модуля f_{13}. При этом d_{15} размещено в станции 5 (см. табл.), а f_{13} выполняется в станции 6. Запись 2-12 означает, что d_2 принадлежит множеству X^0 и размещено в станции 1, а модуль f_{12} выполняется в станции 4.

Из диаграммы следует, что ни одно из выходных данных d_{23}, d_{25}, d_{24}, d_{26} не вычисляется в установленное циклом обновления время. По d_{23} превышение цикла составляет 1 такт, по d_{25} – 0,5 такта, по d_{24} – 3,5 такта, по d_{26} – 3 такта. Это означает, что в таких условиях система работать не может. Здесь следует отметить, что построенная диаграмма отражает простейшую организацию выполнения ГПД, при которой единственным условием запуска модуля является наличие входных данных и свободного процессора, на который распределен модуль. Это с одной стороны не соответствует реальным условиям функционирования СРВ, а с другой – побуждает разработчика к принятию решений о необходимости увеличения ресурсов вычислительной системы либо внесения изменений в программную нагрузку, в распределение модулей и данных по станциям, а также уточнения оценок времени завершения процессов с учетом условий их запуска и взаимодействия. Некоторые из этих решений непосредственно затрагивают концепцию построения СРВ в целом. В первую очередь это относится к ограничениям реального времени, которые отражают динамику функционирования объекта управления.

3. Применение элементов конвейерной схемы при выполнении ГПД

Рассматривая динамику выполнения ГПД в виде совокупности параллельных процессов, следует различать ограничения на время выполнения процессов $\tau(p)$ и ограничения по условиям поступления входных и обновления выходных данных для ПФ. Относительно ПФ величины $\delta(d)$ для входов и $\mu(d)$ для выходов в общем случае являются независимыми и могут не совпадать. Например, для ПФ, содержащей модули f_4,f_8,f_{12},f_{17}, циклы поступления входов $\delta_{Ц}(d_6)=\delta_{Ц}(d_7)=4$, а цикл обновления выхода $\mu_{Ц}(d_{26})=12$.

В данном случае в результате согласования условий поступления входов и обновления выходов принимается, что цикл выполнения ПФ составляет 12 тактов, в течении которых модули f_4 и f_8 выполняются 3 раза, формируя 3 состояния позиции d_{17}, а модули f_{12},f_{17} один раз. Суммарное время выполнения модулей f_{12} и f_{17} составляет 11 тактов, поэтому они могут использовать только состояния позиции d_{17}, полученные ранее в предыдущем цикле. Эта ситуация отражена на временной диаграмме, представленной на рис. 4. Время выполнения ПФ при этом значительно превышает 12 тактов, но за счет использования конвейерной схемы параллельного выполнения модулей ПФ на 2-й и 4-й станциях обеспечивается обновление выхода d_{26} через 12 тактов.

Приемлемость такого варианта динамики работы ПФ оценивает разработчик проекта и при необходимости им могут быть предприняты попытки сокращения времени выполнения ПФ, например, за счет распараллеливания алгоритмов выполнения модулей f_{12} и f_{17}. В этом случае алгоритмы модулей разбиваются на более мелкие фрагменты с последующим распределением их по станциям и производится оценка нового варианта выполнения ПФ. Аналогично рассчитываются результаты применения конвейерной схемы при выполнении других ПФ и ГПД в целом.

Диаграмма на рис. 4 отражает работу ГПД на интервале времени в 16 тактов и включает один полный цикл моделирования равный 12 тактам.

Выделение совокупности процессов и распределение модулей и данных по станциям сохранены прежними. Модули в каждой станции выполняются в соответствии с циклами поступления входных данных, запускающих соответствующие процессы.

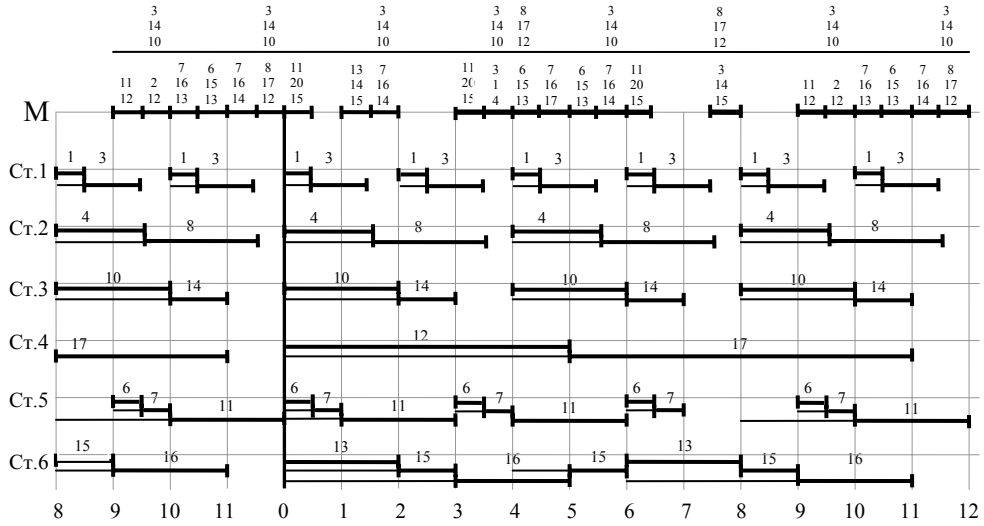

Рис. 4. *Временная диаграмма, построенная с элементами конвейерной схемы выполнения ГПД*

Последовательность выполнения модулей на одной станции производится по условию наличия данных. Если это условие соблюдается для нескольких модулей, то из них выбирается тот, который принадлежит процессу с наименьшим циклом запуска.

В ряде случаев, для установления необходимого порядка выполнения модулей, среди них назначаются относительные приоритеты $\pi(f_m)$. Так, например, в 5-й станции для устранения неопределенности при установлении очередности выполнения модулей f_6 и f_7 назначены приоритеты $\pi(f_6)>\pi(f_7)$. В 6-й станции очередность между модулями f_{15} и f_{16} принимается в пользу f_{15}, т. к. цикл запуска 7-го процесса составляет 4 такта, а 9-го процесса – 6 тактов. При этом следует различать момент запуска модуля и момент начала его выполнения. Все модули, принадлежащие одному процессу, запускаются в момент его запуска, а момент начала выполнения каждого модуля определяется наличием входных данных, освобождением процессора и приоритетами. Так, на 6-й станции (рис. 4) модуль f_{15} запускается с циклом 4, а модули f_{13} и f_{16} с циклом 6. Следовательно, по условию наличия данных модуль f_{13} выполняется первым, а вторым выполняется модуль f_{15}, который принадлежит 7-му процессу с циклом запуска меньшим, чем у 9-го процесса, содержащего модуль f_{16}.

Правила передачи данных и их селекции здесь не рассматриваются. Принимается, что эти правила обеспечивают поступление данных к началу выполнения соответствующего модуля. Например, модуль f_{13} при 2-ом запуске (7-й и 8-й такт) получает данные d_{15}, d_{16}, сформированные модулями f_6 и f_7 соответственно. Эти данные передаются от 5-й станции к 6-й станции в 5-ом и 6-ом тактах. При этом в 5-й станции к моменту запуска f_{13} будет сформировано по два состояния данных d_{15} и d_{16}. Наличие правила селекции в данном конкретном случае указывает – какие из двух состояний данного d_{15} и двух состояний данного d_{16} использует модуль f_{13}.

Время передачи любых данных по магистрали, также как и при построении диаграммы на рис. 3, принимается равным 0,5 такта. Правила записи моментов передачи данных также сохранены. Из диаграммы видно, что и в этих условиях магистраль не справляется с передачей данных и, следовательно, структура вычислительной системы с одной магистралью в данном случае не подходит.

Здесь не следует придавать значение тому, что время передачи данных принято условным, без учета реальных объемов. Это сделано для упрощения построения диаграммы и улучшения ее восприятия при анализе циклов выполнения ПФ. Важным является сам факт наличия возможности с помощью предложенной методики оценить согласованность вариантов вычислительной системы и программной нагрузки. В рассматриваемом примере, по-видимому, можно использовать структуру системы с двумя магистралями так, чтобы ряд передач данных выполнялся параллельно. Для выполнения исследований по выбору структуры сети можно воспользоваться методами, изложенными в [5].

Выводы

1. Анализируя расчеты, связанные с выполнением программной нагрузки, можно констатировать, что предложенная методика позволяет приближенно оценить способность выбранной совокупности станций выполнить программную нагрузку. Если на каком-либо из этапов проверки становится очевидно, что вычислительная система не справляется с программной нагрузкой, то уже на данной стадии исследований могут приниматься решения по изменению модели системы.

2. Конвейерную схему организации работы станций, примененную для построения диаграммы и анализа времени выполнения прикладных функций, следует рассматривать как граничную допустимую возможность организации выполнения графа потока данных с соблюдением заданного времени обновления выходов прикладных функций. Анализ диаграммы позволяет разработчику проекта относительно каждой прикладных функций выявить наиболее значительные задержки времени ее выполнения и принять решения по снижению таких задержек.
3. Исследования конвейерной схемы организации выполнения графа потока данных с целью сокращения запаздывания выполнения прикладных функций позволит в последующем в большей мере формализовать связи между величиной запаздывания и параметрами, определяющими архитектуру вычислительной системы, распределение модулей и данных по станциям, условия динамики работы графа потока данных, условия распараллеливания, селекции состояний, приоритетности.

СПИСОК ЛИТЕРАТУРЫ

1. Погребной В.К. Визуальный уровень представления алгоритмов функционирования распределенных систем реального времени на языке структурного моделирования // Известия Томского политехнического университета. – 2009. – Т. 314. – № 5. – С. 140–146.
2. Погребной А.В. Определение числа и топологии размещения станций многопроцессорной вычислительной системы // Известия Томского политехнического университета. – 2006. – Т. 309. – № 7. – С. 160–164.
3. Погребной В.К., Погребной А.В., Погребной Д.В. Отображение условий динамики функционирования объекта управления на модель системы реального времени // Известия Томского политехнического университета. – 2009. – Т. 315. – № 5. – С. 18–22.
4. Шенброт И.М., Алиев В.М. Проектирование вычислительных систем распределенных АСУ ТП. – М.: Энергоатомиздат, 1989. – 88 с.
5. Погребной А.В., Погребной Д.В. Проектирование структуры локальной сети для распределенной вычислительной системы реального времени // Известия Томского политехнического университета. – 2007. – Т. 311. – № 5. – С. 97–101.

Поступила 05.11.2009 г.

УДК 004.031.6

РАЗРАБОТКА ВСТРОЕННОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ ПРОГРАММИРУЕМЫХ ЛОГИЧЕСКИХ КОНТРОЛЛЕРОВ, ИСПОЛЬЗУЕМЫХ В ОБЛАСТИ ПРОМЫШЛЕННОЙ АВТОМАТИЗАЦИИ

К.С. Щербаков, С.А. Щербаков, В.А. Кочегуров

Томский политехнический университет
E-mail: konstantin.sherbakov@elesy.ru

Представлен метод разработки встроенного программного обеспечения для программируемых логических контроллеров. Метод основан на преобразовании критических участков кода в критические секции, с возможностью целостного исполнения кода процессов.

Ключевые слова:
Разграничение доступа, критический участок и критическая секция, целостное исполнение кода программ.

Key words:
Access isolation, critical region and critical section, entire performance of program code.

Для обеспечения управления технологическими процессами в промышленности и другими сложными технологическими объектами используются, так называемые, программируемые логические контроллеры (ПЛК). Основная работа ПЛК сводится к сбору сигналов от датчиков, их обработке прикладной программой пользователя и выдаче управляющих сигналов для исполнительных устройств [1]. Функциональность ПЛК зависит не только от аппаратного, но и от программного обеспечения (ПО), которое все чаще называют встроенным программным обеспечением (ВПО или Embedded Software). В настоящее время именно к ВПО, как к неотъемлемой части любого ПЛК, предъявляются новые и более жесткие требования, связанные с устойчивостью, многофункциональностью и надежностью работы ПЛК в целом.

Разработка ВПО зачастую затруднена следующими ограничениями, обусловленными особенностями архитектуры микроконтроллеров, на базе которых разрабатывается сам ПЛК и спецификой задач, решаемых ПЛК, а именно: