СЕКЦИЯ 4
ВЫСОКОПРОИЗВОДИТЕЛЬНЫЕ ВЫЧИСЛЕНИЯ

ПОТОКОВЫЕ ДИАГРАММЫ АСИНХРОННЫХ ТЕМПОРАЛЬНЫХ ВЫЧИСЛЕНИЙ ДЛЯ МОДЕЛИРОВАНИЯ И РВ-ВЕРИФИКАЦИИ ПРИЛОЖЕНИЙ РЕАЛЬНОГО ВРЕМЕНИ

А.В. Баландин

Самарский государственный аэрокосмический университет имени академика С.П. Королёва (национальный исследовательский университет) (СГАУ), Самара, Россия

Вводятся в рассмотрение потоковые диаграммы для спецификации и моделирования асинхронных распределённых вычислений над потоками датированных данных в режиме реального времени. В потоковых диаграммах определены понятия: ячейка темпоральной памяти, узел обработки потоковых датированных данных, РВ-верификация темпоральных вычислений. РВ-верификации заключается в мониторинге валидности потоковых датированных данных в ячейках темпоральной памяти в режиме реального времени. Для РВ-верификации необходима натурная реализация потоковой диаграммы на выбранном языке программирования в заданной ОС реального времени. В результате РВ-верификации выявляются ячейки памяти, в которых в результате вычислений возникают датированные данные с низким коэффициентом валидности, что свидетельствует об их несоответствии реальному времени. По итогам мониторинга делаются выводы о возможности и частоте возникновения ошибок темпоральных вычислений в ячейках темпоральной памяти приложения, специфицированного потоковой диаграммой, в заданных вычислительных ресурсах.

Ключевые слова: системы реального времени, режим реального времени, потоковые диаграммы, датированные данные, темпоральная память, валидность датированных данных, темпоральные вычисления, РВ-верификация темпоральных вычислений.

Введение

Приложения реального времени составляют особый класс программных систем, используемых для автоматизации обработки данных. К ним, в частности, относятся многочисленные системы промышленной автоматизации; робототехнические устройства, работающие с машинным зрением; различные виды современного медицинского оборудования, реагирующие на изменение состояния пациента; встроенные системы управления, например, летающими объектами и т.д. Все подобные системы обрабатывают информацию, поступающую во времени от многочисленных «источников» данных, а результаты обработки используются конечными «потребителями» в режиме реального времени. Поэтому приём и обработка данных, их доставка до конечного потребителя осуществляются циклически, а промежуточные и конечные результаты вычислений являются актуальными в течение ограниченных периодов времени. Это является характерной особенностью приложений реального времени.

Почти все данные в приложениях реального времени, так или иначе связаны с различными контролируемыми во времени событиями. Данные от источников поступают в приложение в виде параллельных входных потоков датированных данных, а результаты их

обработки так же доставляются потребителям в виде параллельных выходных потоков датированных данных. Вычисления, которые выполняются над потоками датированных данных в таких системах, естественно должны быть согласованы во времени. Традиционно для согласования параллельных вычислений во времени используют технологию синхронных вычислений, когда все параллельные вычисления осуществляются в рамках одного общего временного такта (такт часов реального времени). Однако реализация синхронных вычислений в распределённых вычислительных системах вызывает значительные, а иногда и непреодолимые затруднения. Поэтому в качестве альтернативы синхронным вычислениям предлагается концепция асинхронных вычислений, в основе которой лежит разработанная автором математическая модель асинхронных темпоральных вычислений над потоками датированных данных [1].

Многочисленные работы по параллельным вычислениям, методам разработки и верификации программных системам реального времени посвящены методам верификации, основанным на темпоральной логике, и оставляют без внимания ошибки использования в вычислениях данных не согласованных во времени. Поэтому предлагаемыми в настоящее время методами тестирования и верификации приложений реального времени такие ошибки обнаружить невозможно. В итоге «отлаженные» программные системы, которые «работают правильно», могут годами сохранять тонкие ошибки вычислений над несогласованными во времени датированными данными, способные крайне негативно проявиться в непредвиденных ситуациях. Поэтому для обнаружения таких ошибок требуется разработка специальных методов верификации, способных выявлять факты использования в вычислениях несогласованных во времени датированных данных. В данной работе такие методы верификации приложений реального времени называются методами РВ-верификации. Формальное определение понятия РВ-верификации введено в [2].

Принципиальная особенность РВ-верификации в том, что она не может быть реализована сугубо теоретически в «бумажном» проекте. Для её проведения требуется натурное или имитационное исполнение приложения в вычислительной системе с реальными характеристиками производительности и именно в реальном времени. Это связано с тем, что при изменении производительности вычислительной системы без изменения самого приложения в общем случае результаты РВ-верификации могут меняться на противоположные. И наоборот, можно пытаться путём эквивалентных преобразований структуры реализуемых в приложения параллельных вычислений добиваться положительных результатов РВ-верификации при неизменных характеристиках вычислительной системы.

В данной работе предлагается метод потоковых диаграмм для спецификации асинхронных темпоральных вычислений и натурного моделирования процессно-нитевой структуры разрабатываемого приложения реального времени, а также метод РВ-верификации, позволяющий в процессе разработки, тестирования и опытной эксплуатации программной системы выявлять ошибки вычислений над не согласованными во времени датированными данными.

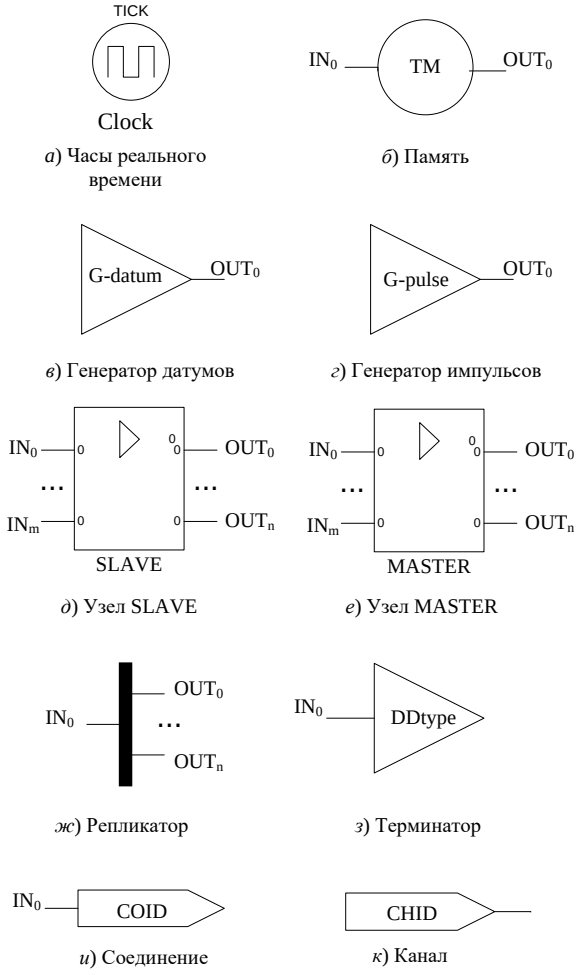
Потоковые диаграммы асинхронных темпоральных вычислений

Использование концепции потоковой обработки данных (flow-based programming - FBP) при программировании приложений реального времени стало настолько же естественным, как, скажем, объектно-ориентированное программирование (ООР). На практике при

проектировании программных систем используются многочисленные вариации потоковых диаграмм, в которых обмен потоковыми данными между параллельными процессами обработки данных осуществляется посредством передачи сообщений. Достаточно сослаться на язык UML, который рекомендуют как стандарт проектирования и разработки программных систем, и его расширения для проектирования систем реального времени, распределенных и параллельных приложений [3]. Однако ни одна из предлагаемых в настоящее время потоковых диаграмм не подходит для спецификации и РВ-верификации темпоральных вычислений. Поэтому в данной работе вводится в рассмотрение оригинальный вариант использования потоковых диаграмм, специально предназначенный для спецификации асинхронных темпоральных вычислений над потоками датированных данных в режиме реального времени, а также для моделирования и РВ-верификации разрабатываемых приложений.

В качестве конструктивных элементов построения потоковых диаграмм используются графические элементы, представленные на рис. 1.

Часы реального времени (Clock) – элемент диаграммы (рис. 1а), моделирующий течение реального интервального времени в тиках, TICK – параметр часов, величина которого задаётся вещественным числом, выражающим величину тика количеством секунд с дробной частью. Часы показывают время в тиках, прошедшее с момента начала работы натурной или имитационной модели потоковой диаграммы.



а) Часы реального времени
б) Память
в) Генератор датумов
г) Генератор импульсов
д) Узел SLAVE
е) Узел MASTER
ж) Репликатор
з) Терминатор
и) Соединение
к) Канал

Рис. 1. Элементы потоковых диаграмм

Память – абстрактная «ячейка» темпоральнй памяти (рис. 1*б*). Память характеризуется размером в байтах, соответствующим максимальному размеру типа потоковых данных, помещаемых в память, и двоичным флагом обновления содержимого. При занесении данного в память оно автоматически датируется меткой времени *t*, соответствующей текущему показанию часов Clock потоковой диаграммы. Флаг обновления принимает значение true. При извлечении данного из памяти флаг обновления сбрасывается в false.

Генератор датумов (G-datum) специализированный узел генерации данных (рис. 1*в*), связывается в диаграмме с одной выходной ячейкой памяти и используется как источник циклически генерируемых через каждый интервал времени *l* потоковых данных – *датумов*, датируемых при помещении в выходную ячейку памяти. Интервал *l* является характеристикой генератора, задаваемой в тиках Clock. . Валидность датумов во времени зависит от заданного для генератора датумов значения *l*, определяющего величину интервала времени, по истечении которого генератор датумов генерирует и заносит в выходную ячейку следующее потоковое данное.

Генератор импульсов (G-pulse) специализированный узел генерации данных (рис. 1*г*), связывается в диаграмме с одной выходной ячейкой памяти и используется в диаграмме как источник событийных потоковых данных – *импульсов*, помещаемых в ячейку выходной памяти. Валидность событийных данных во времени зависит от заданного для генератора импульсов значения *l*, определяющего интервал времени, в течение которого событийное данное должно быть обработано и получен результат обработки.

Порождённые генераторами датумы и импульсы, сохраняются в ячейках выходной памяти как датированные данные [1] - $<\dot{t},l,\nu(\dot{t})\equiv 1>$; $\dot{t}=t$ – показания Clock в момент занесения данного в выходную ячейку темпоральнй памяти, l – интервал репрезентативности данного в тиках Clock (заданная характеристика генератора), $\nu(\dot{t})$ - значение коэффициента валидности порождённого генератором датированного данного, тождественно равно 1. Флаг обновления ячейки темпоральной памяти устанавливается в true.

Генератор получает данные в результате выполнения соответствующей прикладной процедуры, которая обычно осуществляет взаимодействие с внешними по отношению к диаграмме процессами-серверами (например, с сервером базы данных, драйвером внешних устройств, интерфейсом полевых сетей связи с объектами управления и т.п.).

Узел-SLAVE/MASTER – абстрактный вычислитель над потоками датированных данных (рис. 1*д, е*). Характеризуется наборами связей с входными и выходными ячейками памяти и режимом срабатывания SLAVE или MASTER. Узел SLAVE ожидает обновления данного в ячейке входной памяти, присоединённой к входу IN_0 узла, срабатывает, когда флаг обновления станет *true*. Узел MASTER срабатывает, как и генератор G-datum, - циклически через каждый интервал времени *l* – характеристика узла MASTER. При срабатывании узел извлекает датированные данные из входных ячеек памяти, устанавливает в ячейках входной памяти флаг обновления в *false*, запускает назначенную узлу прикладную процедуру обработки данных. Процедура «знает» тип передаваемых ей на обработку данных. Результаты обработки возвращаются узлу. Узел помещает их в соответствующие ячейки выходной памяти в виде датированных данных с характеристикой $<\dot{t},l,\nu(\dot{t})>$, $\dot{t}$ - текущее показание системных часов,. При этом все датированные выходные данные получают одинаковые интервалы репрезентативности *l* и коэффициенты ва-

лидности $\nu(\dot{t})$. Интервал репрезентативности, формируемый узлом *SLAVE*, берётся равным интервалу репрезентативности входного датированного данного, взятого из входной ячейки памяти, присоединённой к входу IN_0 узла *SLAVE*. Интервал репрезентативности, формируемый узлом *MASTER*, берётся равным интервалу циклического срабатывания узла *MASTE*. Коэффициент валидности $\nu(t)$ для всех выходных датированных данных берётся вычисляется по формуле:

$$\nu_{OUT_j}(\dot{t}) = \min_{i=0,m}\{\nu_{IN_i}(\dot{t})\}, \quad j = \overline{0,n};$$

это - минимальная валидность среди валидностей всех полученных узлом входных датированных данных в момент их извлечения из ячеек входной темпоральной памяти. Валидность датированного данного в момент времени его извлечения из ячейки входной темпоральной памяти формируется автоматически и вычисляется по формуле [1]:

$$\nu_i(t) = \begin{cases} \nu_i(\dot{t}_i), t \le \dot{t}_i + l_i; \\ \nu_i(\dot{t}_i)\dfrac{l_i}{t - \dot{t}_i}, t > \dot{t}_i + l_i. \end{cases}$$

В зависимости от семантики формируемых выходных результатов узел может сочетать в себе свойства генератора датумов или импульсов. Если результат срабатывания узла безусловно выдаётся в выходную ячейку памяти, то узел на выходе работает как генератор датумов. Если выдача результата срабатывания узла зависит от выполнения некоторого условия (например, значение результата должно быть больше установленной величины), то узел на выходе проявляет себя как генератор импульсов. Однако узел не может одновременно порождать на выходе и датумы и импульсы.

Репликатор - специализированный узел *SLAVE* (рис. 1*ж*), связанный с одной входной ячейкой темпоральной памяти, и с более чем одной ячейкой выходной темпоральной памяти. Используется в диаграмме для получения в выходных ячейках памяти копий датированного данного и его характеристик во времени, извлекаемого из входной ячейки памяти.

Терминатор – специализированный узел *SLAVE/MASTER* (рис. 1*з*), связанный с одной входной ячейкой памяти. Терминатор извлекает датированное данное из ячейки входной памяти и передаёт их прикладной процедуре обработки данных. Рассматривается в диаграмме как поглотитель входных датированных потоковых данных. Прикладная процедура терминатора «знает» тип данных, которые ей передаются на обработку. Обычно прикладная процедура осуществляет взаимодействие с внешними по отношению к диаграмме процессами (например, с сервером базы данных, системой отображения информации, драйверами внешних устройств, с исполнительными устройствами или устройствами отображения данных в полевых сетях и т.п.). Терминатор может работать как в режиме *SLAVE*, так и *MASTER*.

Соединение – специализированный узел (рис. 1*и*), связанный только с одной ячейкой входной памяти, используемый в диаграмме для передачи датированных данных между смежными узлами, оказавшимся в распределённых фрагментах диаграммы и не имеющих прямого доступа к разделяемым ячейкам темпоральной памяти. Характеризуется значением *COID* (*ID* соединения с каналом). Датированное данное, извлекаемое из вход-

ной темпоральной памяти, передаётся соединением в смежный канал, соответствующий ID соединению, в виде сообщения. Работает в режиме *SLAVE*.

Канал – специализированный узел (рис. 1*к*), связанный только с одной ячейкой выходной памяти, используемый в диаграмме для приёма сообщений с датированными данными и занесения их в выходную темпоральную память. Характеризуется значением *CHID* (*ID* канала). Датированное данное, принятое каналом, помещается в выходную ячейку памяти. Канал вместе с соответствующим соединением обеспечивает в диаграмме репликацию содержимого смежных ячеек распределённой темпоральной памяти. Канал ждёт сообщения, посылаемые соединениями, и срабатывает при их поступлении.

В качестве прикладных процедур каналы и соединения реализуют процедуры передачи сообщений в соответствующих вычислительных сетях.

Натурное моделирование и РВ-верификация потоковых диаграмм

Для РВ-верификации используется натурного программная модель потоковой диаграммы, загружаемая в реальную вычислительную систему. Под натурной моделью понимается почти полностью реализованное в соответствии с полученной диаграммой программное приложение на выбранном языке программирования и в выбранной операционной системе. Из разработанных объектно-ориентированных программных средств строится программный «скелет» приложения. В таком виде приложение способно моделировать режимы генерации потоков датированных данных и темпоральные свойства результатов вычислений над датированными данными в узлах обработки. Для моделирования выполнения прикладных процедур в узлах используется конечный цикл неких «пустых вычислений», в которых количество циклов характеризует «тяжесть» вычислений. Таким образом, получение натурной модели выражается в сборке приложения в соответствии с потоковой диаграммой из программных объектов. Единственное, что откладывается на потом - программирование реальных прикладных процедур. После того, как натурная модель построена, приложение может быть загружено в вычислительную среду и запущено на выполнение для РВ-верификации.

РВ-верификация заключается в оперативном сборе, накоплении и сохранении при натурных испытаниях сведений о появлении в ячейках темпоральной памяти датированных данных с коэффициентом валидности меньше 1. Это признак нарушения режима реального времени. Манипулируя параметрами приложения, такими как приоритет нитей, дисциплина диспетчеризации нитей, количество пустых циклов в прикладных процедурах, можно спланировать эксперименты с натурной моделью для достижения требований режима реального времени.

Пример спецификации, натурного моделирования и РВ-верификации приложения реального времени

В качестве иллюстрации применения диаграмм и РВ-верификации приложений реального времени рассмотрим пример спецификации простой гипотетической системы реального времени, осуществляющей управление наполнением резервуара водой до заданного уровня. Вода подаётся в резервуар с заданной скоростью. Система должна управлять за-

движкой, которая регулирует скорость подачи воды и в итоге - полностью перекрывает подачу. Вид диаграммы приведён на рис. 2.

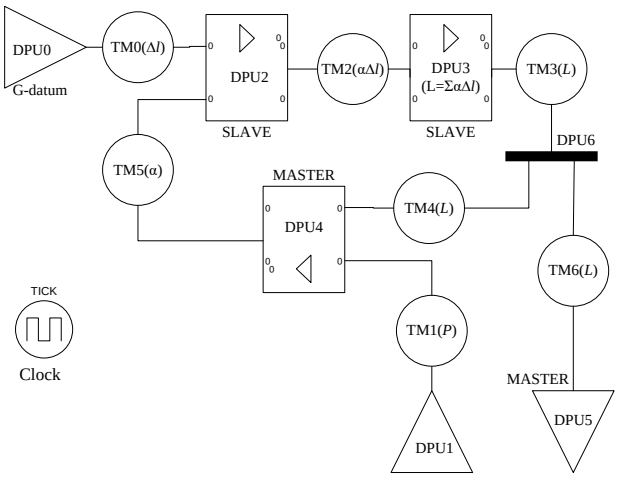


Рис. 2. Потоковая диаграмма приложения реального времени

Генератор датумов DPU0 моделирует подачу воды порциями по Δl, обновляя через каждый тик системного времени содержимое ячейки TM0. Узел DPU2 в режиме *SLAVE* управляет скоростью подачи воды, срабатывая при обновлении ячейки TM0. При срабатывании выполняется процедура вычисления значения $\alpha\Delta l$, результат помещается в ячейку TM2. Узел DPU3 вычисляет значение текущего уровня L, моделируя наполнение резервуара водой. Срабатывает в режиме *SLAVE*, выполняя процедуру суммирования значений, выбираемых из ячейки TM2. Результат помещается в ячейку TM3 и реплицируется репликатором DPU6 в ячейки TM4 и TM6. Терминатор DPU5 в режиме MASTER циклически отображает содержимое ячейки TM6. Генератор DPU1 генерирует импульс со значением P и бесконечным интервалом репрезентативности. Значение P – константа, предельный уровень резервуара, который событийно задаётся извне. Узел DPU4 в режиме MASTER циклически вычисляет коэффициент редукции $\alpha.=(P-L)/P$. Когда $\alpha.=0$, подача воды в резервуар прекращается.

Натурное моделирование скелета приложения было выполнено на одном компьютере в среде ОС реального времени QNX6 (Neutrino) [**Ошибка! Источник ссылки не найден.**]. Распределения фрагментов потоковой диаграммы по разным сетевым узлам не потребовалось. Поэтому такие элементы как «соединение» и «канал» в диаграмме не понадобились.

Приложение было сформировано в виде одного процесса, в котором узлы являлись параллельно выполняемыми нитями. После получения исполняемого модуля была выполнена РВ-верификация натурной модели приложения. Цель РВ-верификации была сформулирована как анализ поведения приложения в случае прекращения подачи воды в систему. Для этого в процедуре генератора DPU0 была предусмотрена ситуация остановки и ячейка TM0 переставала обновляться. В результате в процессе выполнения приложения в ячейках памяти наблюдалось постепенное уменьшение коэффициента валидности хранимых датированных данных (во всех ячейках памяти кроме TM1 данные устаревали). Когда подача воды возобновлялась, валидность данных в ячейках восстанавливалась до 1. Изменяя в приложении точность системных часов, можно было наблюдать изменение погрешности наполнения резервуара. Когда резервуар переставал наполняться, приложе-

ние продолжало функционировать «в холостую». Но когда в реальном времени генератор DPU1 асинхронно изменял значение уровня *P*, то система вновь доводила воду в резервуаре до уровня, доливая или сливая лишнюю.

Заключение

Предложенные в работе потоковые диаграммы асинхронных темпоральных вычислений для спецификации и РВ-верификации позволяют сертифицировать разрабатываемую программную систему как приложение реального времени. Подобные диаграммы должны входить в набор средств проектирования и разработки приложений реального времени, например в UML. Для несложных, например встраиваемых приложений, может быть вполне достаточным выполнить проектирование и разработку программы только с использованием предложенных потоковых диаграмм.

Литература

1. Баландин, А.В. Модель параллельных и асинхронных темпоральных вычислений с автовалидацией // Перспективные информационные технологии (ПИТ 2015). – Том 2: труды международной научно-технической конференции / под ред. С.А. Прохорова. - Самара: Изд-во Самарского научного центра РАН. 2015. – С.3-7.
2. Баландин, А.В., Николаев, А.В. Метод структуризации и РВ-верификации приложений реального времени для систем промышленной автоматизации // Надежность и качество. - Труды международного симпозиума.–Пенза: Изд-во Пенз. гос. ун-та. 2003. – С.378-380.
3. Гома, Х. UML. Проектирование систем реального времени, параллельных и распределенных приложений: Пер. с англ. – М.: ДМК Пресс, 2011. – 704 с.: ил. (Серия «Объектнооориентированные технологии в программировании»).
4. Кёртен, Р. Введение в QNX Neutrino 2. Руководство для разработчиков приложений реального времени. – СПб.: БХВ-Петербург, 2005. – 400 с.