Всеволод Шахов
ОАО “ДНПП” (г.Долгопрудный)

Программно-аппаратная декомпозиция реактивных параллельных систем.

В статье рассматривается метод выявления подзадач в программно-аппаратных системах, имеющих множество параллельно выполняемых однотипных процессов. Данная статья является продолжением моей статьи “**Моделирование программно-аппаратных реактивных систем раскрашенными сетями Петри**”.

Развитие микроконтроллеров в части увеличения производительности, расширение их периферийных возможностей и уменьшения стоимости, позволило разработчикам активно использовать их в своих проектах, создавая программно-аппаратные комплексы, обеспечивающие высокую функциональность за счет программной составляющей.

Обычно в этих устройствах используется один микроконтроллер, на внешнее адресное пространство или порты ввода-вывода которого «навешивается» ряд периферийных модулей, выполняющих роль связующего звена между управляющей логикой и внешней средой. Такая схема является вполне приемлемой для простых (не параллельных, не многоканальных) устройств.

Реактивные параллельные системы, задача которых состоит в реагировании на различные (часто не планируемые) виды событий или сигналы, накладывают ряд требований, связанных как с обеспечением выполнения функций в режиме реального времени, так и с управлением независимыми или частично зависимыми процессами. Решение таких проблем на одном микроконтроллере приводит к следующим неблагоприятным факторам при разработке программного обеспечения для них:

- плохая прогнозируемость временной характеристики при многозадачной обработке;
- ограниченные ресурсы микроконтроллера повышают трудоемкость разработки многозадачных систем;
- наличие большого числа внешних событий и требуемая для них соответствующая обработка множества параллельных задач значительно увеличивает появление трудно выявляемых ошибок. Обычно не учитываются все комбинации внешних воздействий на систему.

Одним из вариантов, облегчающим проектирование реактивных параллельных систем и улучшающих их эксплуатационные качества (ремонтопригодность, отказоустойчивость и пр.) является выделение функционально завершенных минимально зависимых подзадач с реализацией в виде параллельно выполняемых программно-аппаратных компонент, управляемых центральным диспетчером.

Такая декомпозиция предусматривает проведение анализа системы на стадии проектирования, где основными инструментами могут стать современные средства моделирования – язык UML (Unified Modelling Language - универсальный язык моделирования) и раскрашенные сети Петри. С помощью этих средств, можно проводить декомпозицию системы: выявлять параллельно выполняемые процессы, представлять их в виде слабо зависимых между собой программно-аппаратных компонент, моделировать процесс координации и синхронизации обмена данными.

В качестве примера (см. рисунок 1) рассмотрим несколько абстрагированную от конкретного приложения систему, которая содержит минимальное число обрабатываемых данных при достаточном условии возможных проблем, встречающихся при проектировании реактивных параллельных управляющих систем.

Допустим, имеется несколько одинаковых устройств (или механизмов, или каналов передачи данных), которые по специальному временному алгоритму управления включаются в работу, при этом для их работы задействуется ограниченное количество источников питания (или специальных приводов, или схем проверки линий связи). Кроме того, все управление

системой осуществляется от единственного управляющего канала (например, пульта оператора или центрального компьютера).

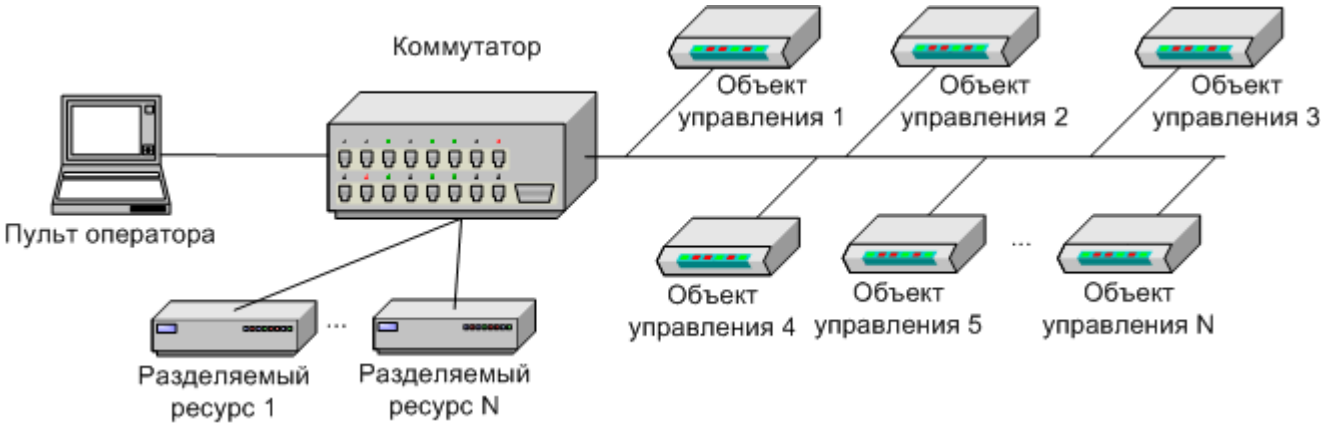

Рис.1 Пример программно-аппаратной системы с разделяемыми ресурсами.

Примерная упрощенная диаграмма кооперации компонентов системы на языке UML представлена на рисунке 2.

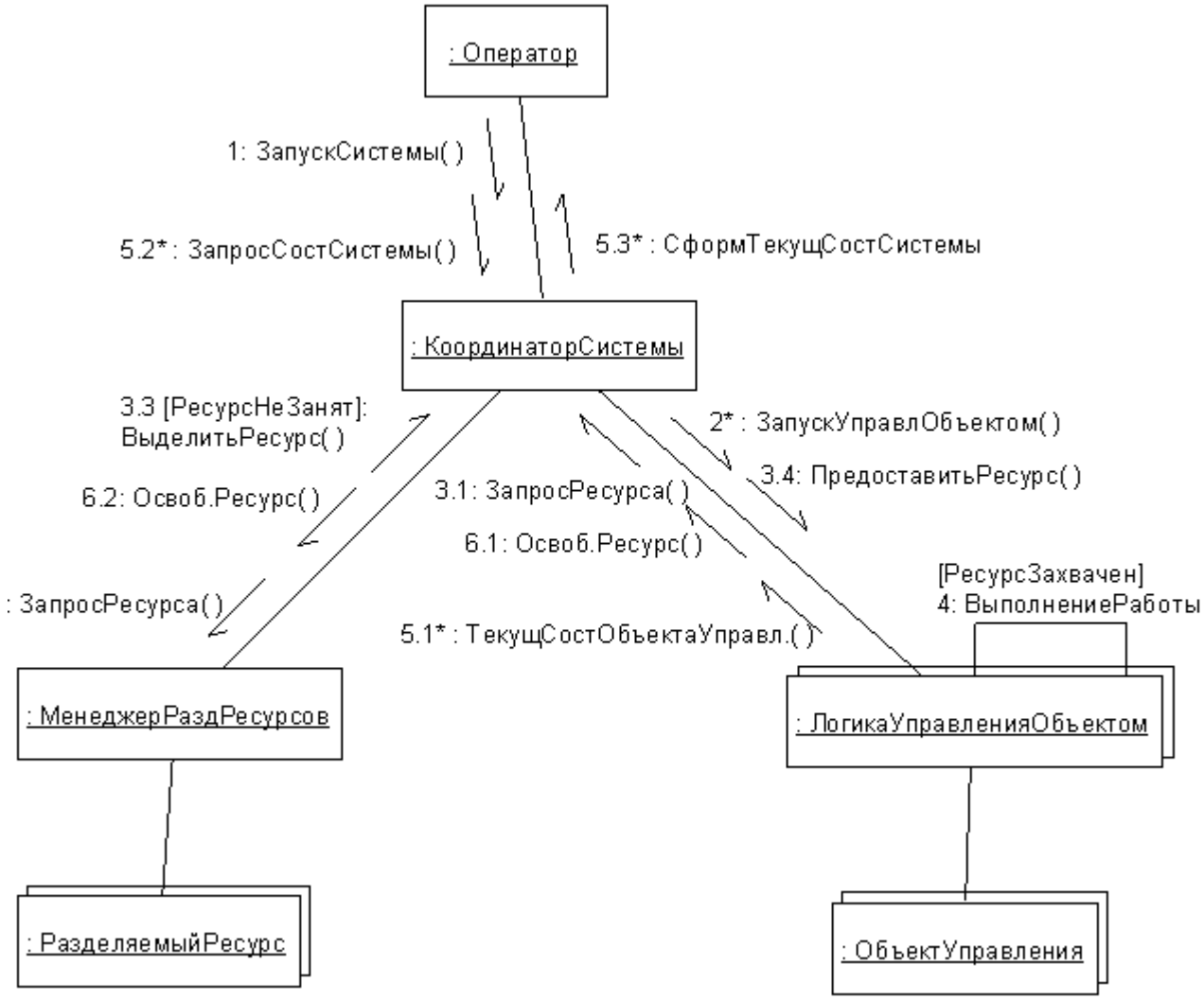

Рис.2 Упрощенная диаграмма кооперации примера реактивной параллельной системы.

Как видно из диаграммы, в первом приближении, просматривается архитектура системы с центральным управляющим элементом (Координатор), связывающим воедино все остальные компоненты, путем ретрансляции межкомпонентных операций. Так, например, запрос на

выделение необходимого разделяемого ресурса, активная задача “ЛогикаУправленияОбъектом” осуществляет через задачу “Координатор”, который “обращается за помощью” к задаче “МенеджерРаздРесурсов”, владеющей необходимой информацией о состоянии ресурсов. Описание приведено в терминах объектно-ориентированного программирования для акцентирования внимания на высокоуровневом рассмотрении вопроса.

Фактически, в примере охвачен ряд задач, встречающихся в параллельных системах:

- Параллельное (псевдопараллельное) выполнение процессов.
- Разделение ограниченных ресурсов.
- Синхронизация данных между различными компонентами системы.

В то же время, в реактивных системах, основная проблема заключается в обработке данных в режиме реального времени. Одним из вариантов описания на языке UML является диаграмма состояний с пометками на дугах о временных условиях. Но ограничение на описание только одного процесса не способствует прозрачному анализу параллельных систем в плане отсутствия наглядного представления процедуры синхронизации процессов.

Введем в пример временные ограничения на работу параллельных задач. Допустим, для запуска каждого устройства требуется воспользоваться разделяемыми (коммутируемыми между устройствами) на время 1с источниками питания, после чего перейти на отработку внутренней временной диаграммы, выполняющейся в течение интервала времени 1..3с. При этом необходимо вести постоянное информирование оператора о процессе выполнения диаграмм во всех устройствах системы. Заключительной стадией является использование разделяемого блока контроля на время 0,5с.

Для конкретизации примера примем число однотипных устройств равным 3, число разделяемых источников питания – 2, число блоков контроля – 1.

Диаграмма работы трех устройств может быть представлена следующим образом (Рисунок 3):

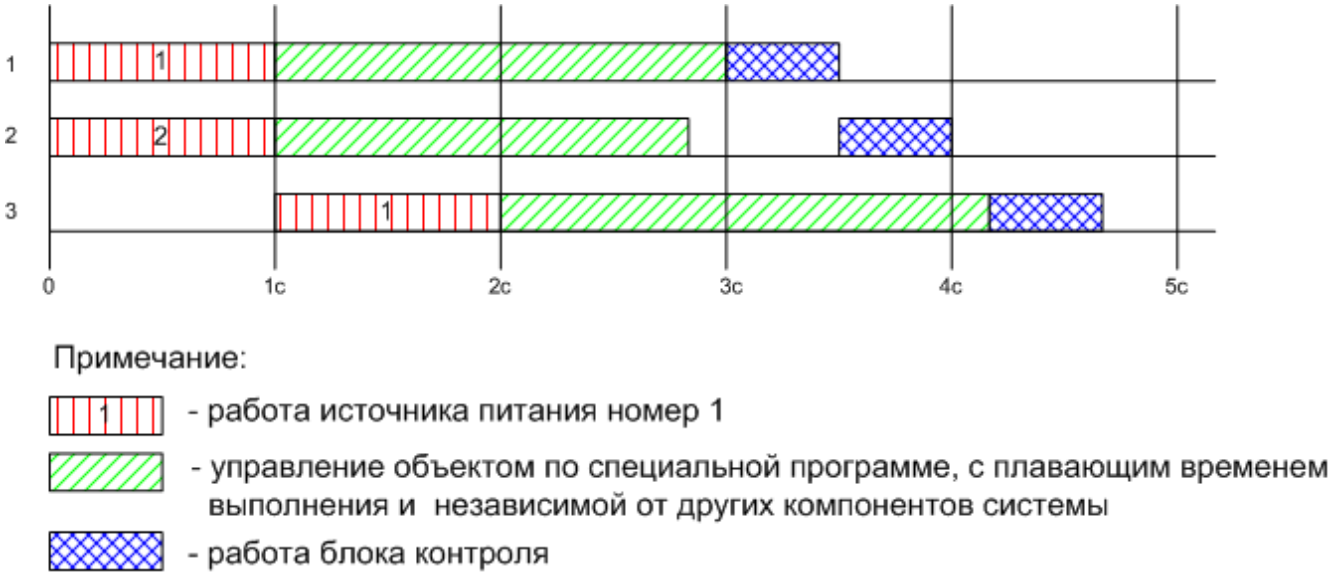

Рис.3 Диаграмма работы трех объектов управления.

Для усложнения примера, введена “плавающая” (не планируемая по времени) внутренняя работа центрального устройства (координатора), которая зависит от внешних факторов (параллельно работающих подустройств). При реализации систем на одном микроконтроллере, особенно, если присутствует интенсивное управление подустройством на данном промежутке времени, именно этот факт становится камнем преткновения при распараллеливании обработки.

Для описания работы параллельных систем все более популярным инструментом становятся раскрашенные сети Петри. На рисунке 4 приведена сеть Петри для описанного выше примера без конкретизации процессов управления, необходимых для работы подустройств.

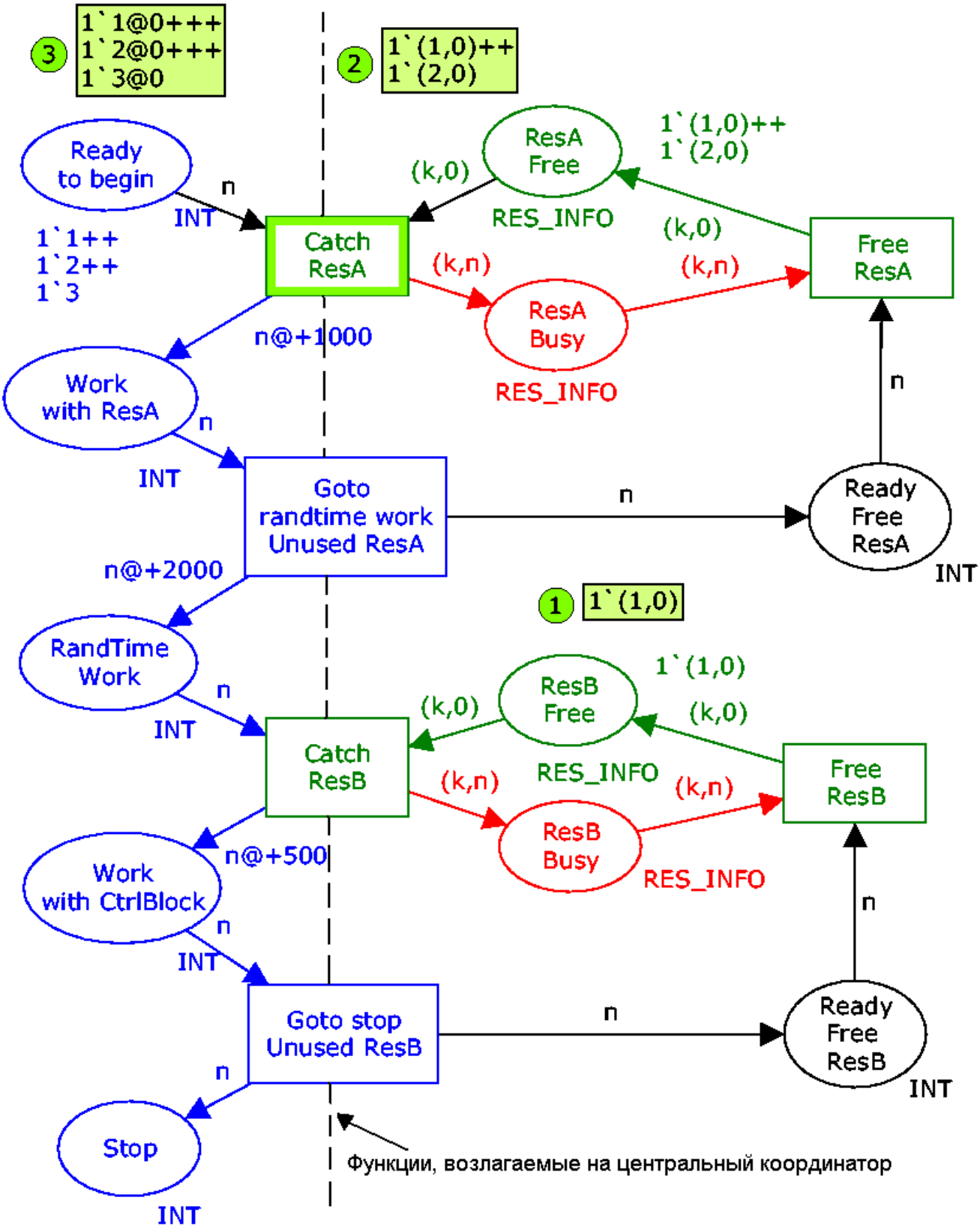

Рис.4 Раскрашенная сеть Петри работы одного объекта управления.

Как можно заметить, эта сеть содержит два типа фишек, одни показывают этапы работы параллельных процессов, другие – состояние разделяемых этими процессами ресурсов. В результате такого разделения фишек и типов позиций легко выявляется подсистема общего (центрального) координатора и его задачи, основной из которых является обеспечение индикации занятости разделяемых ресурсов для процессов, пытающихся их использовать, а также реализация монопольного захвата свободного ресурса конкретным процессом.

Представленная упрощенная сеть Петри не отражает факт непредсказуемости времени работы процесса в позиции RandTimeWork, ее задача показать предполагаемый “срез” системы между компонентами. Таким образом, переходы, связанные с захватом ресурсов (CatchResA, CatchResB) и освобождением ресурсов (UnusedResA, UnusedResB) “трансформируются” в интерфейсные сообщения.

Наглядное выявление функций позволяет разбить систему на подзадачи, реализуемые на программно-аппаратном уровне, независимо друг от друга, естественно, при соблюдении взаимодействия между собой посредством согласованных правил. Применяя декомпозицию системы на аппаратном уровне встает вопрос обеспечения связи между компонентами. В настоящее время, современные микроконтроллеры снабжаются достаточным набором внешних интерфейсов, очень “ценными” из которых являются последовательные, работающие в режиме “ведущий-ведомый” (SPI, I^2C и др.).

Для решения задачи управления обменом сообщениями обычно используют централизованный метод, в котором роль ведущего выполняет центральный элемент системы, управляющий передачей сообщений между подчиненными компонентами через себя. Он по очереди посылает абонентам сообщения, на которые те отвечают подтверждением приема (в случае передачи им данных), либо сообщением с данными (если ожидается прием). При этом методе исключаются конфликты между компонентами, и гарантируется время доступа. Недостатком может служить недостаточная гибкость в случае повышенной потребности в обмене одного из компонентов.

В заключение отмечу, что при проектировании ядра системы стараются всячески абстрагироваться от конкретики, придать ему свойство универсальности с целью обеспечения легкости при дальнейшем добавлении новых функций. В данном примере, координатору отводится роль центрального элемента, производящего обмен сообщениями между управляющими компонентами для выявления потребности в использовании разделяемых ресурсов и предоставлении им общей информации. Фактически, он становится платформенно-независимым транслятором команд-сообщений.

Литература.

1. Гома Х. UML.Проектирование систем реального времени, параллельных и распределенных приложений. Пер.с англ. - М.ДМК Пресс 2002 704 с.
2. Питерсон Дж. Теория сетей Петри и моделирование систем. - М.:Мир,1984.
3. Шалыто А.А. Алгоритмизация и программирование для систем логического управления и "реактивных" систем. - "Автоматика и телемеханика",2000,№1,с.3-39.
4. Шахов В.Ю. Моделирование программно-аппаратных “реактивных” систем раскрашенными сетями Петри. – www.softcraft.ru
5. Jensen K. Introduction to the practical use of coloured Petri nets -\\ URL:http://www.daimi.au.dk/~kjensen/
6. Kristensen Lars M., Christensen S., Jensen K. The practitioner's guide to Coloured Petri Nets - Springer-Verlag,1998.
7. Netjes M, etc. Analysis of resource-constrained processes with Coloured Petri Nets - Eindhoven University of Technology, Netherlands.