ПЕРСПЕКТИВЫ ФОРМАЛИЗАЦИИ МЕТОДОВ ПРОЕКТИРОВАНИЯ ВСТРОЕННЫХ СИСТЕМ

Алексей Платунов, к.т.н., директор, ООО «ЛМТ», г. Санкт-Петербург;
Николай Постников, к.т.н., ведущий специалист, ООО «ЛМТ», г. Санкт-Петербург

Проблема проектирования вычислительных систем до сих пор стоит на одном из первых мест при производстве вычислительной техники. В значительной мере искусственное деление вычислительной системы на аппаратную и программную составляющие и, более того, стремление выделить программную индустрию в отдельную область лишь углубляют кризис.

Как пример решения этой задачи сегодня можно привести процесс комплексного проектирования встроенных систем (embedded systems), которое обычно выполняется компактной командой разработчиков. Стремительное развитие программируемой элементной базы таких систем (микроконтроллеры, ПЛИС, программируемые системы на кристалле) расширяет возможности проектировщика в применении нестандартных решений на всех уровнях организации вычислительной системы. Однако эффективно использовать этот потенциал программисты и аппаратчики смогут, только коренным образом изменив весь процесс создания продукта.

ВВЕДЕНИЕ

Постоянный рост потребности в информационно-управляющих системах различного назначения заставляет разработчиков вычислительной техники активно совершенствовать способы и средства проектирования. Это, несомненно, касается и специализированных (заказных) вычислительных систем, которые непосредственно взаимодействуют с объектом контроля или управления. Такие встроенные системы и контроллерные сети (ВсС) относятся к категории продуктов с доминирующей программной компонентой (SW-dominated products), в основе которых лежит встроенное программное обеспечение (embedded SW). Разработкой такого ПО занимается особый сектор программирования. Специфику встроенного программирования определяют требования работы ВсС в реальном времени, надежности, ограниченности вычислительных ресурсов, энергосбережения, сложности отладки. Более того, технологии встроенного программирования резко отличаются от технологий, используемых в других областях вычислительной техники. Развитие элементной базы и потребность в заказных ВсС выводят встроенное программирование в разряд важнейших технологий проектирования. Область действия встроенного программирования постоянно расширяется, охватывая не только традиционную программируемую элементную базу, но и проектирование аппаратуры в целом.

В настоящее время существует колоссальная потребность во встроенных системах, несмотря на то, что их проектирование характеризуется высокой сложностью. Проблема заключается в существовании огромного количества потенциально пригодных вариантов реализации, соответствующих одному техническому заданию (ТЗ). Эти варианты могут отличаться друг от друга коренным образом, а их предварительная оценка весьма затруднительна. Именно этими фактами, в первую очередь, объясняется рост исследований, направленных на создание технологий эффективного сквозного автоматизированного проектирования ВсС.

Анализ многолетнего опыта разработки микропроцессорных систем различного назначения убеждает в необходимости совместного (параллельного) проектирования аппаратной и программной частей системы, что на практике в рамках традиционных технологий удается реализовать крайне редко. Одной из главных причин такой ситуации следует считать то, что многие разработчики находятся в плену устаревших представлений о схемном проектировании и программировании микропроцессорных систем. Проектирование системы подменяется «почти случайным» выбором микропроцессорной элементной базы (аппаратной платформы) и последующим программным «достраиванием» ее до требований ТЗ. В результате неоправданно усложняется программная часть проекта, в то время как возможности аппаратной элементной базы почти не используются, так как проектирование ведется в ограниченном пространстве микроконтроллерных шаблонных решений.

Понимая современную микропроцессорную элементную базу в широком смысле как совокупность всех видов программируемых элементов (МП, МК, DSP, ПЛИС, конфигурируемые микросхемы памяти, контроллеры интерфейсов и внешних устройств), следует отметить радикально изменившуюся ситуацию в области массового проектирования аппаратных средств ВсС. Из пространства ограниченного набора канонических структур и схемных решений «ранней» микропроцессорной техники проектировщики уже сегодня могут перейти в пространство практически свободного архитектурного проектирования. Диапазон доступных решений в таком пространстве простирается от использования «нерегулярной логики» до создания реконфигурируемых сосредоточенных и распределенных вычислителей значительной сложности.

Такое изменение ситуации на фоне роста спроса на ВсС и требований к ним стимулирует преодоление кризиса проектирования, который проявляется в недостаточной скорости создания и низком качестве продукта. Плачевное состояние в области проектирования ВсС отмечается всеми ведущими специалистами этого сектора информатики и вычислительной техники [1, 2]. Преодолеть же кризис поможет пересмотр методов и технологий всего процесса проектирования в этой области.

ПРОЕКТИРОВАНИЕ ЗАКАЗНЫХ МИКРОПРОЦЕССОРНЫХ СИСТЕМ

Сложившаяся практика проектирования ВсС заключается в выборе

одной из канонических вычислительных платформ (ВП), на базе которой за счет программной надстройки решается прикладная задача. Проект делится на две части: сначала выбирается база (платформа), затем она достраивается вверх (за счет программирования) до получения требуемой ВсС. Такой способ проектирования опирается на отработанные технологические приемы и инструментальные средства, например, языки программирования, на которых описывается конечная задача, исполнительные устройства (готовые вычислительные машины) и трансляторы.

Применяется и другой вариант: выбирается ВП и наряду с достройкой вверх выполняется модификация вниз. В этом случае базовая платформа выступает и в роли прототипа. Такой способ используется реже из-за высокой трудоемкости.

Первая проблема, с которой сталкиваются разработчики ВсС, состоит в следующем:

– существующие языки программирования предполагают описание задачи для идеализированной виртуальной (языковой) машины;

– транслятор отображает эту языковую машину на реальную машину, внося определенные ограничения и не учитывая многие важные технические особенности исполнительской машины (например, особенности системы ввода-вывода, защитных механизмов и др.);

– для того чтобы учесть эти ограничения и особенности, программисту необходимо помимо знаний о языковой машине иметь знания о трансляторе и об исполнительской машине.

На сегодня отсутствует единая система описания этих трех составляющих, они описываются в различных предметных пространствах и языковых стилях.

Вторая проблема – большое число задач, особенно в области систем управления физическими объектами, которые плохо укладываются в схему реализации на основе канонических ВП с языковой программно-реализуемой надстройкой. В результате решения оказываются экономически неоптимальными, либо задача вообще не решается в рамках современных технических средств. В этих случаях необходимо проектировать специализированные ВП, например, с высокой степенью параллелизма и специализацией операционных блоков. Попытка проектировать ВсС на такой архитектурной основе по описанной выше массовой традиционной технологии катастрофическим образом обостряет проблему нестыковки специфических требований языковой машины, трансляторов и ВП. Если для определенного класса задач традиционная схема проектирования еще приемлема, то для специализированных систем эффективность проектирования может сводиться к нулю (огромные проблемы с параллелизмом, с защитными механизмами, с ограничениями реального времени, с тестированием и отладкой и так далее).

Третья проблема связана с постоянно растущим объемом необходимого проектирования ВсС. Разобщенность описаний, отмеченная выше, препятствует повторному использованию разработанных компонентов (аппаратных блоков, программ, реализаций алгоритмов). Магистральным направлением в повышении эффективности проектирования, как отмечают ведущие специалисты, является закрепление результатов разработок в виде абстрактных технических решений, инвариантных к способу реализации. Однако переход в проектировании ВсС на такой уровень сдерживается отсутствием соответствующей языковой базы и инструментальных средств.

Решение указанных проблем возможно через создание единого пространства проектирования ВсС в терминах вычислительных абстракций (вычислительных механизмов, функциональных преобразователей, виртуальных машин, архитектурных агрегатов) и, соответственно, новой модели процесса проектирования. Условиями создания такого пространства являются:

– унификация трактовок hardware/software и процессов их создания;

– представление проектируемой ВсС в рамках возможно большего количества шагов проектирования (от начала) «в чисто вычислительном базисе» (в терминах вычислительных абстракций);

– поиск решений по организации ВсС в проектном пространстве с координатами HW/SW, препроцессирование либо процессирование, вычислительные либо не вычислительные аспекты организации системы;

– использование «аспектной технологии», которая позволяет явно проследить трансформации ключевых составляющих ВсС по ходу проектирования (функциональность, инструментальный, надежностный аспект и другие) и учесть в процессе проектирования дополнительные факторы. Речь идет о факторах, которые явно (или вообще) не присутствуют в ТЗ, но серьезно влияют на результаты проектирования (инструментальный аспект, возможности коллектива, доступные технологии, учет задела и интересов коллектива, и другие).

Важным является и тезис «платформно-ориентированного» проектирования, который формирует гибкий *взгляд на понятие ВП как обобщенной виртуальной машины, зафиксированной для создания очередной функциональной надстройки*.

Указанные пути решения опираются на расширенную трактовку понятия «элементная база», куда входят объекты всех уровней абстракции, принадлежащие различным составляющим аспектной технологии проектирования.

ОСНОВЫ АСПЕКТНОЙ ТЕХНОЛОГИИ ПРОЕКТИРОВАНИЯ ВСТРОЕННЫХ СИСТЕМ

Сегодня аспектный подход (Aspect-Oriented Software Development) оформился как самостоятельное направление в программировании систем различного назначения (www.aosd.net), которое существует наряду с другими технологиями. Авторы считают перспективным распространение аспектной идеологии и на область проектирования ВсС.

Центральным понятием процесса проектирования является *архитектура ВсС – совокупность концептуальных аспектов ВсС некоторого уровня детализации, адекватно отображающая проектируемую систему для данного уровня рассмотрения*. В перечень концептуальных аспектов, составляющих архитектуру ВсС, помимо традиционных структурных и функциональных элементов должны войти надежностный, конструктивно-технологический, энергетический, условий эксплуатации, инструментальный, повторного использования, организационно-экономический, документный и другие. Полный перечень аспектов, конечный для конкретного процесса проектирования ВсС, определяется конкретным ТЗ в сочетании с опытом и целями коллектива разработчиков.

Рассмотрим два объекта верхнего уровня аспектной технологии, с которыми имеет дело разработчик: *архитектура проекта* (framework) и *архитектура продукта* (проектируемой ВсС). Эти архитектурные

представления содержат все составляющие соответственно процесса проектирования и создаваемого продукта. Назовем такие составляющие *аспектами проектирования* (или просто аспектами).

Начиная с некоторого уровня интегральной сложности проектируемой системы, все более заметной становится потребность описания такой системы в терминах технических решений и логики их взаимодействия. Разработчики оказываются перед сложной дилеммой: с одной стороны, необходимо уменьшать уровень детализации представления системы на верхних уровнях, с другой стороны — это может привести к потере важных характеристик (например, габариты или энергопотребление). То есть, понижая уровень детализации в представлении системы, разработчик должен уверенно контролировать все ее аспекты.

Дополнительные проблемы при проектировании современных ВсС создает значительная разнородность аспектов, которые следует учесть. Например, для таких аспектов, как габариты, энергопотребление и объем памяти (для микросхем памяти) достаточно сложно вывести единую зависимость, сводящую эти характеристики к «универсальным» единицам измерения. Если проводить поиск подходящего устройства среди доступных в настоящий момент на рынке, то очевидно, что с течением времени и развитием технологий множество точек в пространстве [габарит × энергопотребление × объем], представляющих собой допустимую комбинацию, заметно переместится.

Также важным моментом при проектировании сложных ВсС является возможность их масштабирования или повторного использования в том или ином виде. При огромной скорости смены элементной базы и лавинообразном росте доступных ресурсов нерационально разрабатывать систему, опираясь исключительно на тот или иной микроконтроллер (или даже семейство микроконтроллеров), микросхему программируемой логики и интерфейс передачи данных. Во многих случаях использование той или иной элементной базы требуется в ТЗ на разрабатываемую систему, но даже тогда использование конкретного элемента должно быть всего лишь *ограничением*, но ни в коем случае не *базой* при разработке системы.

Основным подходом в проектировании системы становится *построение* сложной, разноплановой, полнофункциональной *модели* системы с учетом заранее определенных ограничений и последующим отображением модели на ту или иную элементную базу.

Разноплановое, охватывающее многие (либо все) аспекты системы представление будем называть *архитектурным* представлением системы. Отдельные элементы такого представления назовем *агрегатами*, т.к. они объединяют совершенно разнотипные аспекты представления системы. Таким образом, *архитектурный агрегат* (АА) представляет собой базовый элемент процесса проектирования системы. АА могут иметь различную степень абстракции, представлять собой технические решения или конкретные вычислительные узлы, определять разное количество аспектов для описываемого элемента. Модель системы, выраженную в терминах АА, будем называть *архитектурной моделью* или *А-моделью* системы.

Каждый из аспектов характеризуется примитивами соответствующего множества, внутренней в рамках аспекта логикой работы модели, а также внешней логикой взаимодействия с другими аспектами.

Основываясь на аспектной полноте АА в составе конкретной архитектурной модели, определяются три типа моделей и указывается их место в проектировании ВсС. Архитектурная модель может быть абстрактной, виртуальной и реализуемой.

Абстрактная А-модель. В такой модели существует хотя бы один абстрактный АА. Модель принципиально нереализуема и требует дальнейшей доработки, если это модель конкретной целевой системы. Но у таких моделей есть свой способ применения, а именно, такие модели следует рассматривать как платформы для построения конкретных систем. Абстрактными А-моделями, перешедшими в разряд общепринятых платформ, можно считать стандартные интерфейсы (USART, I^2C), шины (PC104, PCI, USB), протоколы (CANopen, TCP/IP), вычислительные ядра (MCS51, x86, PowerPC, JAVA), операционные системы (QNX, MS Windows) и многое другое. При этом перечисленные модели описывают различные перечни аспектов, что не мешает им быть общепризнанными стандартами. Некоторые абстрактные А-модели также можно использовать в качестве повторно используемой платформы в рамках коллектива. То есть, конкретный коллектив может зафиксировать определенное удачное решение, обозначить направление разработок, обеспечить преемственность, определив абстрактную А-модель системы и развивая ее в тех или иных конкретных приложениях. Особенно такие модели и локальные (внутри коллектива), повторно используемые платформы интересны с хорошо проработанным инструментальным аспектом. Именно развитие инструментария оправдано при возможности его многократного повторного использования и адаптации под конкретные проекты.

Виртуальная А-модель. В такой модели нет абстрактных АА, но существует хотя бы один виртуальный АА. Такая модель не может быть реализована из-за виртуальных АА, но она уже полностью определена и, в принципе, может быть реализована, если расширить доступную элементную базу. Вообще, такие модели могут быть использованы и как абстрактные, но в большинстве случаев требуется доводить их до реализации. Для этого нужно избавиться от виртуальных АА. Этого добиваются двумя способами: изменением модели или изменением внешних факторов. В первом случае разработчики изменяют модель (проводят процесс проектирования) до тех пор, пока виртуальных АА в ее составе не останется. Во втором случае модель остается неизменной, а меняются внешние факторы. Например, происходит уточнение ТЗ в сторону изменения ограничений; обучение коллектива; переход на другие вычислительные ядра; использование новых для коллектива технологий (использование микросхем программируемой логики, смена языка программирования, поверхностный монтаж и многослойные печатные платы).

Реализуемая А-модель. В данной модели нет ни одного абстрактного или виртуального АА. Такая модель может быть реализована коллективом в доступной ему элементной базе.

В процессе проектирования целевой системы на начальных этапах происходит конкретизация и верификация архитектурной модели. Результатом проработки архитектурной модели становится «золотая» модель. «Золотая» модель — это верифицированная и зафиксированная архитектурная модель системы, не ограничивающая способов реализации.

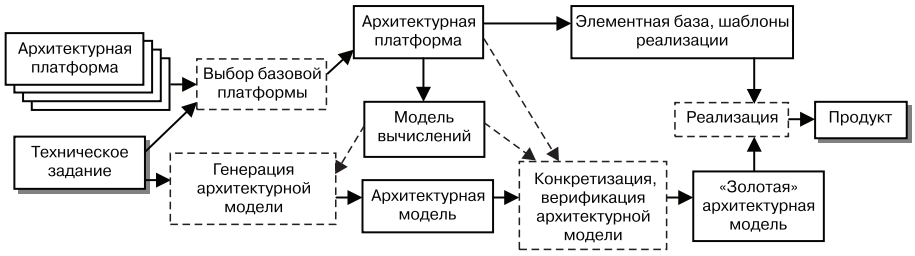

Рис. 1. **Место и значение архитектурной платформы в процессе проектирования ВсС**

Моделирование на начальных этапах проектирования связано с доказательством адекватности разработанной архитектуры начальным требованиям. На этих этапах применяются достаточно сложные методы функциональной верификации, призванные доказать соответствие характеристик архитектурной модели и требований к системе.

При окончательном формировании «золотой» модели процесс моделирования переходит в фазу реализации, и моделирование призвано доказать эквивалентность реализаций и «золотой» модели. На этом этапе преобладают сравнительно простые, но трудоемкие методы эквивалентной верификации и для большинства преобразований существуют специальные САПР.

Еще одной важной задачей «золотой» модели становится создание исходных спецификаций для разработчиков, которые занимаются конечной реализацией компонентов и узлов системы. Являясь полноценной архитектурной моделью, «золотая» модель специфицирует не только непосредственно конечную функциональность, но и такие аспекты, как параметры обеспечения параллельности работы, допустимые пределы энергопотребления, необходимые механизмы надежности, перечень документов, необходимый уровень инструментальной поддержки и многое другое.

Традиционно основными в процессе проектирования ВсС являются структурно-функциональные требования. Они представляют собой *поведенческий аспект* архитектурной модели. В рамках аспектного подхода к процессу проектирования поведенческим аспектом является *модель вычислений*.

В современных условиях разработка каждого проекта «с нуля» неэффективна и практически невозможна, необходимо увеличивать степень повторного использования компонент и решений. Кроме того, в каждом конкретном проекте необходимо производить выбор элементной базы, ведущих аспектов проектирования, внешних ограничений проекта (критериев проектирования). В рамках аспектного рассмотрения процесса проектирования инструментом повторного использования концептуальных решений становится *«архитектурная платформа»*.

Архитектурную платформу следует рассматривать как объединение следующих элементов проектирования:

– аспектное пространство процесса проектирования (перечень аспектов проектирования);

– модель (модели) вычислений;

– внешние факторы, задающие допустимые соотношения между отдельными аспектами (критерии проектирования);

– перечень зафиксированных шаблонов повторного использования;

– элементная база.

Реконфигурируемость платформы определяется как способность к изменению «воплощаемой» при реализации модели вычислений. Надстройка над архитектурной платформой, созданная с целью изменить модель вычислений, называется *операционной средой*. На рисунке 1 показана обобщенная схема процесса проектирования ВсС с использованием архитектурной платформы.

МОДЕЛИ ВЫЧИСЛЕНИЙ, ПЛАТФОРМЫ И ВЫЧИСЛИТЕЛЬНЫЕ МЕХАНИЗМЫ

В основе работы любой вычислительной системы лежит концепция организации вычислительного процесса, которую часто называют вычислительной моделью (model of computation). Эффективность как реализации (проектирования), так и самого процесса решения прикладной задачи в первую очередь определяется выбором вычислительной модели.

Далеко не все известные вычислительные модели имеют примеры удачных (эффективных) реализаций. Кроме того, в большинстве распространенных вычислительных систем соседствуют (иногда не согласованно между собой) многие вычислительные модели, воплощенные в таких их стандартных компонентах, как процессоры, операционные системы, трансляторы, СУБД и другие.

В универсальных ЭВМ или системах на их основе с такой ситуацией можно соглашаться, жертвуя суммарной эффективностью и надежностью вычислений в угоду сокращению затрат на массовую разработку не критичных прикладных систем. В области же информационно-управляющих систем, распределенных систем реального времени и особенно ВсС адекватность, корректное сочетание и эффективная реализация выбранных вычислительных моделей стоят на первом месте в ряду критериев проектирования.

Но как обстоит дело с использованием понятия «модель вычислений» в современной информационной индустрии? К сожалению, в проектировании оно присутствует в основном неявно. Его носителями являются немногие специалисты высшей квалификации (чаще всего они выступают в качестве архитекторов проектов). Однако и среди них лишь часть оперирует данным понятием в формализованном виде, остальные используют его интуитивно. Большинство прочих специалистов не имеют цельного взгляда в этом вопросе, хотя на практике постоянно сталкиваются с вычислительными моделями на примере концепций языков программирования, архитектур процессоров, средств автоматизации проектирования (моделировщики, компиляторы, логические синтезаторы и так далее).

Ситуация усугубляется ограниченным числом публикаций, рассматривающих вычислительную технику через призму первичности и многообразия вычислительных моделей, а также демонстрирующих соответствующие методы проектирования. Те же проблемы присутствуют и в системе подготовки специалистов по вычислительной технике и программированию. Не только сохраняется, но продолжает увеличиваться разрыв в техническом мировоззрении «аппаратчиков» и «программистов». Что касается русскоязычной литературы в данной области, то авторы готовы привести лишь пару ссылок на фундаментальные работы, в которых находит отражение данный взгляд на организацию и проектирование вычислительных систем, однако он изложен «между строк» [3] или прак-

тически не затрагивает «аппаратные» пласты архитектуры [4].

Понятие модели вычислений, как и большинство основополагающих понятий вычислительной техники, четко не определено. Например, авторы [4], многократно используя данный термин в контексте фон Неймановской архитектуры, потоковой архитектуры, архитектур распространенных языковых машин (C, Java), не приводят его определения. В статье [5] *вычислительная модель* определяется *как шаблон взаимодействия компонент, снабженный полезными для моделирования свойствами*. Также в данной статье отмечается неэффективность (зачастую невозможность) восприятия того, как устроена и работает вычислительная система, например, по ее описанию на языке программирования высокого уровня с известной семантикой, если отсутствует ее концептуальное описание (ее вычислительная модель).

Действительно, выразительные возможности одного (формального) языка позволяют зафиксировать в описании проектируемой вычислительной системы (упрощая: программа + компилятор + ОС + процессор) только часть всего предполагаемого вычислительного процесса. Яркий пример такой ситуации — существование знакомых специалистам в области ПЛИС уровней «структурного» и «поведенческого» программирования на VHDL. Рядовой программист в меньшей степени сталкивается с такой неоднозначностью при использовании распространенных языков программирования. Однако и здесь это касается только «тривиального» программирования, например, не затрагивающего параллельные процессы. Таким образом, даже на «бытовом» уровне проектирования специалисты постоянно сталкиваются с проблемой осознания вычислительной модели.

В работе [6] авторы отмечают необходимость использования *однозначно-трактуемого формализма для представления как высокоуровневых спецификаций проекта, так и вариантов его архитектурного решения*. Отмечается, что *использование такого формализма многократно увеличивает эффективность всех этапов проектирования ВсС — специфицирования, формальной верификации, корректной детализации и оптимизации, реализации*. Именно *такой формализм* они *обозначают термином «вычислительная модель»*.

Попробуем пояснить понятие вычислительной модели и ее место в иерархическом представлении вычислительной системы.

Проектирование вычислительной системы как объекта, решающего конкретную задачу пользователя, целесообразно рассматривать как *организацию вычислительного процесса в пространстве и времени в оговоренных техническим заданием ограничениях*. Именно в этом случае *концепция организации вычислительного процесса может рассматриваться как модель вычислений*.

Элементами и инструментами организации вычислительного процесса являются:

— концепции проектирования (платформно-ориентированное, модельно-ориентированное, акторно-ориентированное и другие);

— вычислительные модели (потоков данных, автоматные, дискретных событий и другие);

— платформы (вычислительные, инструментальные, протоколов взаимодействия и так далее);

— механизмы (как *технические решения из различных областей в абстрактном представлении*);

— элементная база (как *множество реализаций механизмов*).

В трактовке понятия вычислительной модели прослеживаются два направления. Если принять за отправную точку в анализе иерархию организации ВсС, то одно направление основывается на желании ограничиться использованием простых (по «принципу действия») моделей одного (самого высокого) уровня; другое направление предполагает формализацию представления виртуальных машин независимо от их сложности и принадлежности (в том числе и текущей) к уровню вычислительной иерархии.

Первое направление не привязано к реализации, оно демонстрирует «рафинированный» взгляд на вычислительный процесс. Примерами такого подхода можно считать модели потоков данных, модели на основе потоков управления, в том числе автоматные модели, модели синхронных процессов и другие.

Второе направление в значительной степени затрагивает реализацию виртуальной машины, «вторгаясь» в область вычислительной архитектуры. В этом случае рассматривается принцип действия того или иного вычислителя. Например, последовательных программных интерпретаторов с принстонской или гарвардской архитектурами, суперскалярных вычислителей. К этой же категории вычислителей следует отнести так называемые языковые машины (C, PicoJAVA, Prolog и другие), которые являются носителями семантики языковых структур данных и управления.

Не менее важно создание неоднородных моделей ВсС, которые включают различные базовые модели вычислений. В этом направлении в мире проводятся активные исследования и уже разработан ряд САПР с такими возможностями.

Необходимо отметить, что предложенный авторами взгляд на два варианта понимания и использования термина «модель вычислений» требует серьезного обсуждения.

Пояснения требует и термин «виртуальная машина», который уже неоднократно использовался выше. Для этого вернемся к понятию вычислительной архитектуры и уточним его.

Архитектура ВсС — представление ВсС с минимальным уровнем детализации, демонстрирующее все ее уникальные технические решения в части организации вычислительного процесса. Такое представление требует *отображения внешней функциональности последовательно на внутреннюю функциональную организацию и внутреннюю структурную организацию ВсС.*

Таким образом, архитектурное представление направлено на раскрытие реализации некоторого вычислителя. Уровень представления определяется минимальным набором известных элементов (вычислительных механизмов), до которых необходимо опуститься в рассмотрении ВсС, чтобы удовлетворить перечисленным выше требованиям. В зависимости от того, кому адресуется данное архитектурное представление ВсС, набор «известных» элементов может существенно различаться.

Под *виртуальной машиной понимается вычислитель, для которого определены правила поведения* (например, система команд, условия ввода команд, данных, получения результата, правила синхронизации процесса), *позволяющие однозначно описать алгоритм решения задачи*. Описание виртуальной машины демонстрирует лишь внешние свойства вычислителя и правила его использования, не касаясь его устройства. Виртуальная машина может иметь различные реализации, например, в виде конкретной ЭВМ или последовательности «ручных» рас-

четов. В этом смысле виртуальная машина – это «черный ящик», обладающий известной функциональностью и внешним интерфейсом.

Однако, как было отмечено выше, без знания концепции организации вычислений, то есть вычислительной модели, на которую ориентирована виртуальная машина, ее использование неэффективно, а часто и невозможно. Принцип выделения виртуальных машин в сегодняшней вычислительной технике – один из наиболее мощных технологических инструментов разработчика, позволяющий структурировать процесс проектирования, обеспечить переносимость и повторное использование наработок, масштабировать технические решения и проекты.

Еще один исключительно важный формализм в вычислительной технике, осмысление и развитие которого происходит в течение последних нескольких лет, – *вычислительная платформа*. Одна из наиболее продуктивных трактовок этого понятия представлена также в [5], где оно определяется как *множество проектов, удовлетворяющее (или вытекающее из) некоторому общему условию*. Примерами таких платформ могут послужить бинарные образы программ под архитектуру x86, байт-коды Java-программ, Wintel персональные компьютеры и многое другое.

Платформа в контексте понятия модели вычислений может рассматриваться как единство «внешнего» и «внутреннего» представления функционально-завершенного и функционально-значимого объекта в составе ВсС. Возможно, следует трактовать вычислительную модель и вычислительную архитектуру как две стороны одного вычислительного объекта. А вычислительную платформу – как интегральное представление такого объекта.

Итак, попробуем привести еще одно определение. Будем называть *вычислительной моделью математическую модель, демонстрирующую пользователю вычислительные возможности вычислителя и правила их использования*.

Несколько слов необходимо сказать о вычислительных механизмах. Основываясь на приведенном выше определении механизма, данная абстракция играет роль основного «кирпичика» в организации вычислительного процесса. В отличие от обсуждавшихся выше цельных в функциональном плане понятий, вычислительный механизм решает частную задачу, например, буферизации данных, декодирования управляющего слова, выборки и анализа команды, кэширования информационного потока. Диапазон функциональной сложности вычислительных механизмов ограничен лишь фантазией разработчика. Различная функциональная направленность механизмов, используемых в вычислительной технике, создает серьезные проблемы для формализации их представления и операций над ними, однако работы в этом направлении ведутся.

ЗАКЛЮЧЕНИЕ

Традиционная практика создания встроенных систем основана на использовании известных разработчику значимых (но все же второстепенных) объектов элементной базы в качестве первичных факторов формировании архитектуры. Такие объекты (тип микропроцессора, операционная система, сетевой интерфейс), выбираемые часто также по вторичным признакам, с одной стороны, навязывают разработчику многие технические и организационные решения, с другой стороны, лишают его возможности рассмотреть альтернативные варианты, получить сбалансированное решение. Можно ли эффективно проектировать встроенные вычислительные системы, не используя обсуждавшиеся выше понятия? Сложные – безусловно, нет.

Конечно, данные рассуждения выходят за рамки «профессионального взгляда» проектировщика-кодера (как программиста, так и аппаратчика), у которого проектирование и отладка (доводка) продукта составляют соответственно 10% и 90% времени. Но здесь остается только сожалеть.

Перечислим главные препятствия на пути развития и внедрения технологий проектирования, основанных на абстракциях вычислительной архитектуры:

– объективная сложность восприятия ВсС на уровне моделей вычисления (исключение составляют проектировщики с базовым математическим образованием);

– отсутствие адекватных и эффективных средств описания, моделирования, синтеза (сегодня существуют САПР, фрагментарно закрывающие такой процесс проектирования);

– кажущаяся неэффективность попыток применения таких методов в условиях, когда нет готовых CASE-средств (то есть слишком много надо думать самому);

– проблемы с подготовкой специалистов в области вычислительной техники, программирования, телекоммуникаций (по факту), когда специализация студента в узкой области не базируется на представлении вычислительной системы с единых позиций организации вычислительного процесса (в результате в практической деятельности выпускники руководствуются следствиями, а не причинами).

Следует напомнить, что внедрение в процесс подготовки специалистов понятий, обсуждаемых в работе, активно происходит в ведущих зарубежных университетах [7].

Формализация и автоматизация процессов проектирования всегда требует колоссальных усилий. Область заказного проектирования встроенных систем – пример, где такие усилия прикладывать необходимо. В заключение можно привести слова из [1]: «Цель состоит в получении эффективного, полуавтоматического, прозрачного, верифицируемого и математически корректного процесса проектирования от исходных спецификаций до реализации для систем с доминирующей программной составляющей, которые создаются на «глубоко-программируемых» платформах».

ЛИТЕРАТУРА

1. Alberto Sangiovanni-Vincentelli and Grant Martin. A Vision for Embedded Software. Proceedings of CASES 2001, Atlanta, Georgia, November, 2001.

2. Technology Roadmap on Software-Intensive Systems: The Vision of ITEA (SOFTEC Project), 2 ed. issued by the ITEA Office, Eindhoven, May 2004. Chapter 5, «Engineering», p. 69–82.

3. Таненбаум Э. Архитектура компьютера (четвертое издание). – СПб.: Питер, 2002. – 704 с.

4. Непейвода Н.Н., Скопин И.Н. Основания программирования. – Москва – Ижевск: Институт компьютерных исследований, 2003, 868 с.

5. E. Lee, S. Neuendorffer, M. Wirthlin. Actor-oriented design of embedded hardware and software systems. Journal of Circuits, Systems, and Computers. 29 p.

6. L. Lavagno, A. Sangiovanni-Vincentelli, E. Sentovich. Models of Computation for Embedded System Design. – 1998 NATO ASI Proceedings on System Synthesis, Il Ciocco (Italy) 1998, 57 p.

7. www.cs.lth.se/home/Krzysztof_Kuchcinski/DES/