УДК 004.436, 004.75

АВТОМАТИЗАЦИЯ РАЗРАБОТКИ ПРОГРАММНЫХ СРЕДСТВ СИСТЕМНОГО УРОВНЯ ДЛЯ СЕТИ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ПРОЦЕССОРОВ ЦИФРОВОЙ ОБРАБОТКИ СИГНАЛОВ

Сидоров С.Б.

ФГБОУ ВПО «Нижегородский государственный технический университет им. Р.Е. Алексеева», Нижний Новгород, Россия, (603950, ГСП-41, Нижний Новгород, ул. Минина, 24), e-mail sidorov@nntu.nnov.ru

В работе предложен подход к решению задачи автоматизации разработки системных программных средств для мультипроцессорных систем цифровой обработки сигналов. Указана область применения рассматриваемого подхода. Разработаны правила описания конфигурации мультипроцессорной системы на основе документа в формате XML. Приведен перечень общих предопределенных свойств элементов описания проекта. Приведен пример описания системы на формализованном языке для конкретной платформы. Реализованы инструментальные средства генерации программного кода на основе описания конфигурации и рассмотрены базовые принципы их работы. Сделаны выводы по использованию рассмотренного подхода для сети процессоров. Обозначены преимущества, достигаемые при использовании автоматизации решения задач системного уровня. Приведены заключения о возможности использования генератора для сети отечественных высокопроизводительных сигнальных процессоров ВПЦОС.

Ключевые слова: мультипроцессорные системы, сигнальный процессор, системные средства, язык описания, генератор кода.

AUTOMATION OF DEVELOPMENT OF SYSTEM SOFTWARE FOR THE NETWORK OF HIGH-PERFORMANCE DIGITAL SIGNAL PROCESSORS

Sidorov S.B.

Nizhny Novgorod State Technical University n.a. R.E. Alekseev, Nizhny Novgorod, Russia, (603950, Nizhny Novgorod, st. Minina, 24), e-mail sidorov@nntu.nnov.ru

An approach to the problem of automation of development of system software for multiprocessor systems of digital signal processing is proposed. The scope of the considered approach is specified. We develop rules for the description of a configuration of multiprocessor system on the basis of XML document. The list of the common predefined properties of elements is provided. In article the example of the description of system in the formalized language for a concrete platform is given. Workbenches of generation of a program on the basis of a configuration are developed and the basic principles of their operation are considered. Conclusions on use of the considered approach for a network of processors are drawn. The advantages reached when using automation of development of system software are designated. The conclusions about possibility of use of the generator for a network of high-performance signal processors are given.

Keywords: multiprocessor systems, signal processor, system tools, description language, code generator.

Для архитектуры современных систем цифровой обработки сигналов характерной чертой является использование нескольких сигнальных процессоров, связанных в сеть посредством коммуникационных интерфейсов. На сети процессоров реализуется распределённая обработка информации, поступающей в общем случае, из нескольких внешних источников [3].

Для широкого круга задач процессорные узлы, в зависимости от своего положения в сети, должны выполнять разные стадии обработки информации, не являющиеся однородными. Как следствие, на каждом таком узле должно быть реализовано свое прикладное программное обеспечение, вид которого определяется конкретной задачей и стадией обработки [2]. В то же время реализация системного программного обеспечения во многом

определяется архитектурой процессорного узла и, как следствие, является однотипной для разных частей системы. Упрощение решения задач системного уровня может быть обеспечено за счет повторного использования ранее разработанных системных библиотек. Однако при подобном подходе для каждого процессора по-прежнему требуется разрабатывать свои управляющие программные модули, отличные по набору используемых аппаратных компонент процессора и по значениям параметров их конфигурации, что приводит к увеличению времени разработки системы в целом [1].

Целью данной работы является рассмотрение автоматической генерации системных программных средств для сети процессоров цифровой обработки сигналов на основе описания всей системы обработки, получаемого в качестве прямого результата выполнения этапа проектирования.

Автоматизация программирования включает комплекс технических, программных, языковых и информационных средств, осуществляющих преобразование описания в программный код устройства [6]. К таким средствам относятся, в том числе входной язык, генератор кода и целевой язык. На рис.1 приведена структурная схема автоматической генерации программного кода.

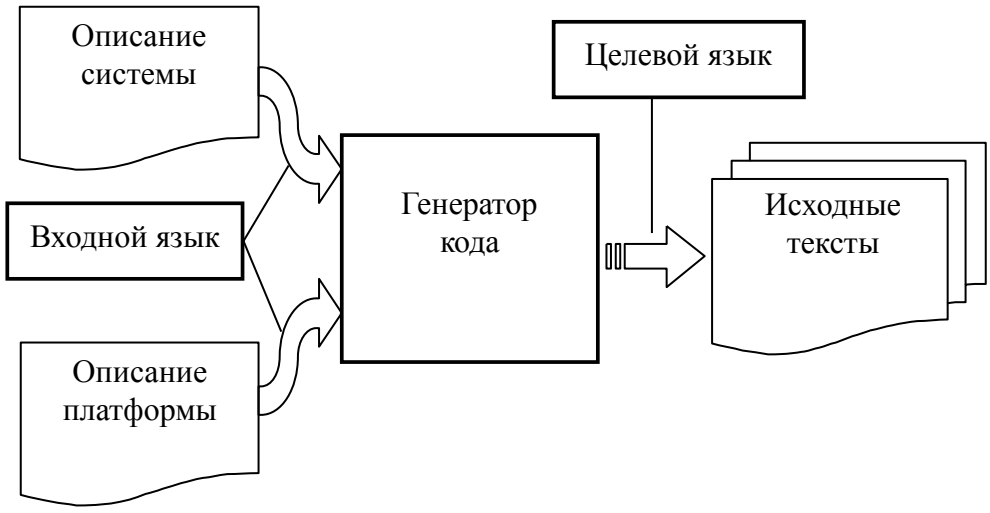

Рис.1. Структурная схема автоматической генерации

Для описания архитектуры разрабатываемой системы используется специальный входной проблемно-ориентированный язык. Он используется для описания исходных данных разрабатываемой системы, технологическим процессом ее обработки и для передачи исходной информации в генератор кода.

Данный принцип не является абсолютно новым, существует ряд решений, среди которых широкую известность приобрел инструментарий DSP/BIOS [7] фирмы Texas Instruments. Однако он ограничивается автоматизацией разработки системного слоя только для одного процессора.

Предполагается, что система цифровой обработки сигналов реализуется на базе комплекса различных модулей, таких как процессорные, АЦП, ЦАП, контроллеры, организованных в сеть.

Генератор применяется для формирования программного кода, исполняемого сигнальными процессорами процессорных модулей. Результатом его работы являются программные компоненты на языке ассемблера, которые совместно с созданными разработчиком функциональными компонентами обеспечивают решение задачи ЦОС.

Область ответственности генератора включает следующие системные задачи:

- инициализация устройств ввода-вывода для обеспечения внутреннего и внешнего взаимодействий процессорного модуля;
- управление внутренними компонентами процессоров (таймерами, DMA-каналами, разделяемой памятью и т.д.);
- обработка прерываний;
- реализация низкоуровневого протокола внутреннего и межмодульного обмена;
- маршрутизация потоков команд и данных между компонентами модулей, как внутри модуля, так и межмодульная (с заданным маршрутом или автоматически определяемым);
- вызов пользовательских обработчиков событий (диспетчер событий).

Для генерации кода используется описание проекта системы на специальном формализованном языке в текстовом виде.

Генерация кода осуществляется на базе описания всего проекта, как совокупности взаимодействующих модулей, а не каждого отдельного модуля независимо. При этом возможно обеспечить автоматизацию проверки корректности описания проекта, в частности:

- отсутствие конфликта имён однотипных компонент (для однозначной идентификации);
- допустимость геометрии расположения модулей;
- полнота описания каждой компоненты;
- допустимость связей между компонентами;
- соответствие протоколов обмена на передающей и принимающей стороне.

Описание проекта содержит полные описания всех компонент, из которых состоит программно-аппаратная система обработки. Основная часть описания представляет собой иерархическую (древовидную) структуру описаний компонент. При этом на одном уровне приводятся последовательные описания компонент одного порядка, а на вложенных уровнях – описания их составляющих частей. Количество возможных вложенных уровней не ограничивается, а определяется уровнем сложности компоненты.

Описание компоненты должно быть полным, то есть должны быть определены значения всех необходимых параметров. При этом если компонент включает в себя ряд необязательных компонентов нижнего уровня, то они могут не определяться, что автоматически означает их отсутствие. При этом описание компоненты всё равно считается полным.

Каждая компонента может быть поименована для ссылки на неё из описаний других компонент. Имя должно быть уникальным как минимум в пределах того уровня и ветки дерева, на котором оно определяется. Однако для компонент разных видов допускается использование одного и того же имени.

Допустимо предположить, что в проекте будут присутствовать однотипные компоненты с одинаковым набором параметров. Сюда можно отнести случаи однородной обработки данных несколькими процессорными модулями, использование множественных каналов обмена с одинаковым протоколом, подобная организация процессорных модулей и т. д. В этом случае во избежание дублирования описания каждой из повторяющихся компонент, в проекте могут быть определены поименованные классы компонент. Класс компоненты представляет собой логическое описание, такое же, как и объекта (описания физической компоненты), с помощью которого задаются значения параметров. Дополнительно, этому описанию назначается уникальное (в области однотипных компонент) имя.

При дальнейшем описании любого объекта в иерархии вместо непосредственного указания значений параметров одним из атрибутов для него может быть указано имя класса. Это приводит к автоматическому назначению параметрам компоненты величин из описания класса. В отличие от описания объекта, определение класса может быть неполным, то есть значения некоторых параметров остаются не определёнными. В этом случае они должны явно определяться при описании объекта. Кроме того, в описании объекта может переопределяться значение параметра, указанного в классе объекта. Это позволяет выполнять адаптацию объектов под конкретное окружение.

Использование классов объектов позволяет приобрести ряд преимуществ. Прежде всего, это позволяет чётче выделить структуру проекта, формируя его набором компонент, имеющих смысловую связь с понятиями предметной области. Также значительно упрощается получение полного описания проекта, путём уменьшения объёма проекта и сокращения требуемого времени на его формирование. Как следствие, ускоряется внесение изменений в повторяющиеся компоненты, поскольку необходимо изменить только значения параметров в классе объектов, а не у каждого независимо.

Перечень допустимых компонент определяется типом проекта. Этот перечень с описанием иерархический отношений компонент и их атрибутов содержится в специальной базе данных

описания платформы. Для каждого типа проекта готовится свое описание платформы. Кроме перечня там же приводятся правила генерации кода для компонент.

Описание проекта приводится на формальном языке, основанном на использовании XML[5]. Для любого типа проекта определены 2 вида компонент:

- project – корневой элемент, включающий описание всего проекта;
- classes – ветвь определения классов проекта (может встречаться произвольное количество раз).

Перечень дополнительных элементов и их свойств определяется типом проекта. Например, ниже представлены некоторые из доступных элементов для проекта типа TigerShark201Multiboard:

- dsp – определение процессорного модуля;
- adc – определение модуля АЦП;
- dac – определение модуля ЦАП;
- link (module) – описание соединения по link-порту модуля;
- cpu (dsp) – описание конфигурации процессора и ПО на нем;
- memory (dsp.cpu, dsp) – описание конфигурации банка памяти;
- timer (dsp.cpu) – описание конфигурации таймера.

Все элементы содержат предопределённый набор свойств. У каждого элемента может быть определён атрибут name. Его интерпретация зависит от того, где располагается описание. Если описание находится внутри элемента classes, то значение атрибута задаёт имя нового класса. В противном случае, это имя описываемого объекта. Также у каждого элемента может быть определён атрибут class. Значение атрибута задаёт класс элемента. Кроме этого любой элемент может быть временно отключен атрибутом disable со значением 1. В этом случае в генерации кода данный элемент не участвует, хотя в проекте информация о нём остаётся.

На одном уровне иерархии может находиться несколько элементов одного вида. Например, в состав процессорного модуля (dsp) может входить несколько процессоров (cpu). Для их различения используется атрибут id. Значение атрибута определяет дескриптор элемента, представленный порядковым номером.

На рис.2 приведен пример фрагмента описания конфигурации системы.

```
<project name= "Filter" platform= "tigersharc-201" >
	<classes>
		<link name="Common"></link>
	</classes>
	<dsp name= "Interface" row="0" column="0">
		<link transmitter="Input" receiver="0" class="Common"></link>
		<link transmitter="0" receiver="Processing" class="Common"></link>
		<link transmitter="Processing" receiver="1" class="Common"></link>
		<link transmitter="1" receiver="Output" class="Common"></link>
		<cpu id="0"></cpu>
		<cpu id="1"></cpu>
	</dsp>
	<dsp name= "Processing" row="1" column="0">
		<link transmitter="Interface" receiver="3" class="Common"></link>
		<link transmitter="3" receiver="2" class="Common"></link>
		<link transmitter="2" receiver="Interface" class="Common"></link>
		<cpu id="2"></cpu>
		<cpu id="3"></cpu>
	</dsp>
	<adc name= "Input" row="0" column="-1">
		<link transmitter="0" receiver="Interface" class="Common"></link>
	</adc>
	<dac name= "Output" row="0" column="1">
		<link transmitter="Interface" receiver="1" class="Common"></link>
	</dac>
</project>
```

Рис. 2. Пример описания конфигурации системы

Перечень элементов, допустимых для конкретного проекта, и их полное описание содержится в файле конфигурации. Описание типа проекта приводится на XML-подобном языке со следующими компонентами:

- root – корневой элемент, включающий описание всего типа проекта;
- element – ветвь описания допустимого вида элемента;
- attr – ветвь описания атрибута (свойства) элемента;
- part – ветвь объявления типа элемента, как допустимого в качестве составной части исходного.

В компоненте element может указываться атрибут software – признак генерации кода для элемента. Например, ветвь объявления процессора должна включать определение этого атрибута software="1". При отсутствии явно заданного значения этого атрибута генератор кода для соответствующего элемента выполняться не будет.

В компоненте part может указываться атрибут idcount – количество компонент данного вида на одном уровне иерархии. Например, если в процессорном модуле располагается до четырех процессоров, то ветвь объявления процессора как составной части должна включать определение атрибута idcount="4".

Генератор кода формирует программу на языке ассемблера, наподобие препроцессора программ на языке С. Используется база предварительно подготовленных шаблонных программных компонент, в текст которых внедрены специальные команды генератора (аналог директив препроцессора).

Обработка компоненты заключается в выполнении генератором команд из самой

компоненты. Фактически выполнение команды заключается в замене одной последовательности символов другой последовательностью (в том числе и пустой). Обработчику компоненты дополнительно передаётся информация об одном или нескольких элементах из описания проекта, необходимая для генерации кода.

Все команды генератора начинаются со специального символа, позволяющего отделить их от просто текста компоненты. Вид символа требует уточнения во избежание конфликта с языком ассемблера. Например, в качестве такового можно использовать @. В частности, рассматриваются следующие команды:

@define имя значение – определить указанное имя, назначив ему заданное значение;

@@имя@ – подставить значение вместо имени;

@undef имя – сделать указанное имя неопределённым;

@include «шаблон» – включить и обработать указанный шаблон;

@ifxxx . . . @endif – группа команд условной обработки. Если условие выполнено, то обрабатывать внутренний блок, иначе исключить его.

Имя представляет собой последовательность латинских букв, цифр, знака подчёркивания. Кроме этого, в качестве имени может использоваться ссылка на атрибут элемента из описания проекта. Значение представлено текстовой строкой, заключённой в двойные кавычки, или может ссылаться на переменную или на атрибут элемента из описания проекта. Таким образом, описание проекта используется в качестве источника значений переменных при генерации кода.

Общепринято считать, что автоматизация какого-либо процесса приводит к некоторому ухудшению качества получаемых результатов по сравнению с «ручным» управлением этого процесса, когда все аспекты полностью находятся под контролем человека. Это, безусловно, справедливо для тех предметных областей, где ключевую роль играет творческое начало и необходимость принимать эвристические решения. Однако, как уже было отмечено выше, решение задач системного уровня определяется заданием корректной конфигурации системы, четкого указания значений многочисленных параметров, ассоциированных с конкретной аппаратной платформой и, таким образом, представляет «рутинную» задачу. Поэтому ее автоматизация не приводит к ухудшению качества получаемых результатов.

Рассмотренный подход был успешно применен для автоматического создания системного программного обеспечения ряда проектов по разработке систем цифровой обработки сигналов с использованием нескольких DSP TigerShark 201s, объединённых в сети различных конфигураций [4]. При этом предварительно однократно была обеспечена поддержка данной платформы, посредством разработки базы данных ее описания.

Также подтверждена возможность использования описанного метода для автоматизации

разработки применительно к перспективному отечественному высокопроизводительному сигнальному процессору ВПЦОС.

Предложенный подход обеспечивает упрощение деятельности специалистов по разработке программного обеспечения, приводит к уменьшению количества ошибок, допускаемых при разработке, повышению надежности программ и, как следствие, к сокращению общего времени, затрачиваемого на решение задачи. Это, в свою очередь, позволяет повысить эффективность реализации сложных программно-аппаратных систем.

Работа выполнена при поддержке Министерства образования и науки в рамках договора № 02.G25.31.0061 от «12» февраля 2013 года (в соответствии с Постановлением Правительства Российской Федерации от 9 апреля 2010 г. № 218).

Список литературы

1. Колесов Ю.Б. Моделирование систем. Динамические и гибридные системы. – СПб.: БХВ-Петербург, 2012. – 224 с.

2. Немнюгин С., Стесик О. Параллельное программирование для многопроцессорных вычислительных систем. – СПб.: БХВ-Петербург, 2002. – 400 с.

3. Олифер В.Г. Компьютерные сети. Принципы, технологии, протоколы. – 4-е изд. – СПб.: Питер, 2014. – 944 с.

4. Программный комплекс управления коммуникационными интерфейсами процессоров TigerSHARC TS201S – Свидетельство о государственной регистрации программ для ЭВМ №2013660715.

5. Extensible Markup Language (XML) 1.1 (Second Edition), [Электронный ресурс] // URL: http://www.w3.org/TR/xml11/ (дата обращения: 27.04.2015).

6. Stahl N., Voelter M. Model-Driven Software Development: Technology, Engineering, Management. – John Wiley&Sons, 2006. – 444 p.

7. TMS320C55x DSP/BIOS 5.x Application Programming Interface (API) Reference Guide - Literature Number: SPRU404Q, August 2012.

Рецензенты:

Хранилов В.П., д.т.н., профессор кафедры компьютерных технологий в проектировании и производстве Нижегородского государственного технического университета имени Р.Е. Алексеева, г. Нижний Новгород;

Ермолаев В.Т., д.т.н., профессор кафедры бионики и статистической радиофизики Нижегородского государственного университета имени Н.И. Лобачевского, г. Нижний Новгород.