УДК 004.318; 004.27; 004.8

Хведчук В.И.

ПОДХОД К ОПИСАНИЮ АРХИТЕКТУРЫ МИКРОПРОЦЕССОРОВ

Введение. На сегодняшний день имеются мощные средства описания и моделирования СБИС, выполненные на базе VHDL [1] или других аналогичных языковых средств. Доступны в свободном доступе в Internet и модели современных семейств аппаратуры, выполненные с их использованием. Для наиболее сложных предлагаются также и соответствующие средства разработки (kit) и программные инструментальные средства. Вместе с тем, имеется необходимость реализации отдельных функций операционных систем (ОС) в аппаратуре. Такого рода устройства уже предлагаются к использованию. Так, в 1985 году была разработана микросхема Intel 80150, реализующая базовые примитивы ОС. Впоследствии эта возможность была частично включена в процессоры типа Pentium.

Необходимость определения процессов ОС присутствует, прежде всего, в антивирусных пакетах. Описание возможно на различных уровнях, включая процессы ядра ОС. При этом возможно замедление основных операций на пользовательском уровне, выполняемых на современных быстродействующих процессорах до уровня 16-разрядных процессоров с тактовой частотой 4 МГц.

Еще одним направлением использования средств описания архитектуры является описание HAL – Hardware Abstraction Layer. Одним из его приложений является определение элементов драйвера. Благодаря последовательному процессу по их стандартизации и модификации выполняется реальная поддержка операционной системы. В наибольшей степени она осуществлена для ОС Windows, в меньшей степени Linux, что и отразилось на их реальной распространенности.

Также традиционным направлением, где используется описание архитектуры микропроцессора, является задача компиляции и обратная ей задача декомпиляции.

В целом, проблема проектирования аппаратурных средств и в современных условиях является актуальной, несмотря на развитие архитектур Intel и др. По-прежнему необходимо преодоление разрыва между аппаратурой и программным обеспечением [2].

При этом, соответствующих специализированных средств моделирования процессов ОС не выявлено. Поэтому возникла задача разработки необходимых средств описания архитектуры микропроцессоров с учетом поддержки ОС. Для поддержки обработки описания, возможно, понадобится соответствующая аппаратурная поддержка.

В рамках данного направления предполагается решение следующих подзадач: 1) разработка языка описания процессов; 2) для поддержки функционирования языка описания процессов необходима разработка структуры средств АРМ моделирования процессов; 3) для ускорения выполнения поиска в описании необходима разработка параллельной модели логического вывода; 4) для увеличения быстродействия параллельной модели необходима разработка ее аппаратной реализации; 5) необходима разработка средств декомпиляции как элемента архитектуры АРМ моделирования процессов для обеспечения построения описания процессов уже имеющихся приложений.

В качестве основы для построения языка описания процессов целесообразно использование HAL/S [3]. Его главные приложения – для встроенных систем реального времени.

1. Язык описания процессов. Для определения связи ресурсов и процессов предлагается описывать процессы в ОС следующим набором:

$$(Func, CPU, MEM, HM),$$

где $Func$ – набор функций процесса; CPU –ресурс центрального процессора; MEM – ресурс оперативной памяти; HM – ресурс внешней памяти.

Для определения информационной зависимости процессов в ОС используется отношение R:

$P_i R P_j$, если для выполнения хотя бы одной из функций $Func_{i,k}$ процесса P_i требуется выполнение хотя бы одной из функций $Func_{j,s}$ процесса P_j.

При этом нижним сечением $R^-(\mathrm{P_i})$ этого отношения будет являться множество всех процессов P_m, зависящих от P_i, и открытых для изменения. Положим $A = R^-(\mathrm{P_i})$, тогда $R_i A M_j$, если хотя бы одному отношению из R_i заданы права доступа процессов M_j.

Для определения и анализа иерархии процессов предлагается использовать язык (R, M, A) описания информационной зависимости R. В этом языке задаются и отношения разграничений процессов M, а также отношения возможности изменений A. Описание имеет декларативный характер.

Задача компиляции рассматривается как набор правил (теорем) для построения исполняемого кода. Задача декомпиляции менее разработана, и, в силу этого, не имеет столь мощной автоматизации, как задача компиляции. Тем не менее, как задача, обратная к задаче компиляции, она также представима набором декларативных правил.

Для поддержки декларативного характера языка (R, M, A), а также инструментов декомпиляции, возможно добавление на аппаратном уровне средств выполнения языка логического программирования.

Возможная архитектура классов для поддержки декларативного описания [4] приведена на рис. 1.

2. Элементы структуры средств АРМ моделирования процессов. В обобщенную архитектуру предлагаемого АРМ (рис. 2), в числе прочих элементов, входят: язык описания иерархии процессов, средства декомпиляции, аппаратные средства поддержки декларативного языка. Для ускорения выполнения поиска при реализации декларативного описания введена модель параллельного логического вывода с соответствующей аппаратной поддержкой. Средства декомпиляции добавлены с целью получения из уже имеющихся готовых приложений описания на языке описания иерархии процессов.

Язык описания иерархии процессов (R, M, A). Используются следующие основные группы функций примитивов процессов: управление заданиями и задачами; управление прерываниями; управление памятью; управление сообщениями между задачами; управление синхронизацией между процессами; управление окружением.

Примерами функций группы управления заданиями и задачами являются: создание задачи; удалить задачу; приостановить задачу; снять задачу.

Функции $Func_{j,s}$ процесса P_j описываются набором своих примитивов $\{N(P_j)\}$. Права доступа назначаются на процесс или на ресурсы.

Введение примитивов необходимо для описания основных элементов системы защиты. Пример примитива:

$$P_j(CPU_j, MEM_j, HM_j) :\!- Func_j\{N(P_j)\}.$$

Отношение R задается в виде отношений: $P_i(CPU_i, MEM_i, HM_i) :\!- P_j(CPU_j, MEM_j, HM_j)$.

Множество A задается как $A(cl(P_i)) :\!- clall(P_i, LP_j)$. Здесь LP_j – список клозов (правил), которые имеют в заголовке одинаковое имя предиката P_j.

Хведчук Владимир Иванович, *к.т.н., доцент Брестского государственного технического университета.*
Беларусь, БрГТУ, 224017, г. Брест, ул. Московская, 267, e-mail: liddan@mail.ru.

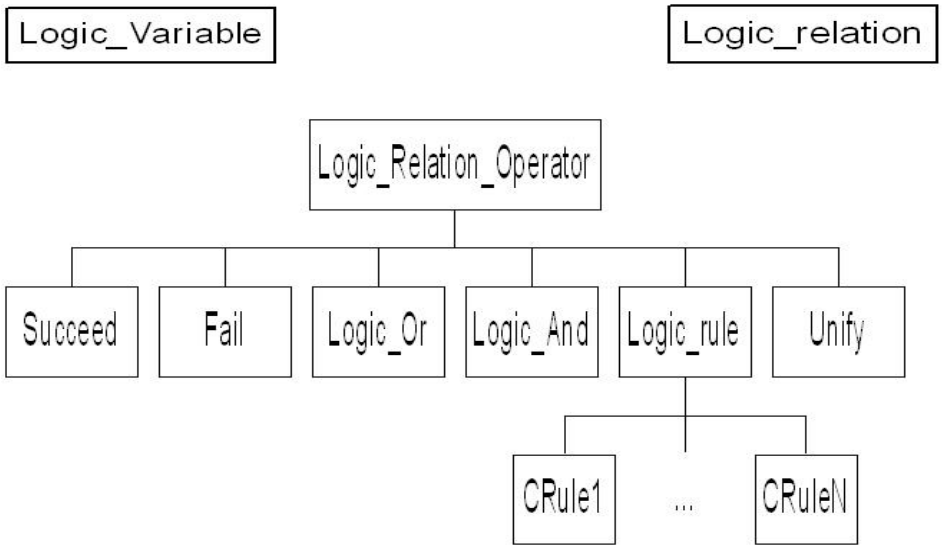

Рис. 1. Архитектура классов для декларативного описания

Рис. 2. Обобщенная архитектура АРМ моделирования процессов

Предикат *clall* является истинным, если истинны все клозы, имеющие в заголовке предикат P_j. Аргумент *cl* является истинным, если является истинным предикат его аргумента.

Средства декомпиляции. Возможно использование, наряду со стандартными средствами декомпиляции, дополнительных средств. Теоремы (правила) компиляции представимы в виде

$$T_z(LA) :- cl(LT_z).$$

Здесь T_z – один из предикатов, формирующих выходной язык;

LA – список его аргументов;

LT_z – список клозов (правил), которые имеют в заголовке одинаковое имя предиката T_z.

По аналогии с построением компилятора возможно построение правил декомпиляции. С целью усиления возможностей декомпилирующих средств и инструментов (R, M, A) вводится поддержка их поддержки аппаратурными средствами. Для ускорения обработки предлагается параллельная реализация логического вывода, в частности операции унификации.

Модель для параллельного логического вывода. При логическом выводе имеет место ориентированный граф Q, в котором

$$P_j^i(P_e)QP_e = Y_j^i(P_e)$$

при условии необходимости унификации предиката $P_j^i(P_e)$ из тела предиката P_e. Здесь P_ζ – предикат запроса. P_j – *j*-ый предикат, одноименный P_ζ $P_j^i(P_e)$ – *j*-ый предикат, находящийся на i-ом уровне унификации (число унификаций после унификации предикатов, одноименных P_ζ) из тела утверждения, в заголовке которого находится предикат P_e, являющийся *e*-м в теле предиката уровня *i*-2; $Y_j^i(P_e)$ – унификация уровня *i* для *j*–го предиката P_e. Существует возможность параллельного выполнения унификаций одного уровня, а также разных уровней *i* и *j* (*i* < *j*), но не имеющих путей перехода по дугам, которые исходят из вершин уровня *i* к вершинам уровня *j*.

Разработано представление декларативного описания. По каждому предикату тела (для факта создается пустой предикат тела) создаются следующие вершины: *ANDFORWARD* – ссылка на следующий предикат тела; *ORBACK* – ссылка на последнюю для данной вершины доказанную альтернативу (по ИЛИ назад);

ANDBACK – ссылка на доказываемый предикат тела; *ORFORWARD* – ссылка на следующую для данной вершины альтернативу (по ИЛИ вперед). При переходе по *ORFORWARD* меняются *ANDFORWARD* и *ORFORWARD*. При переходе по *ANDFORWARD* меняется *ANDFORWARD*. При переходе по *ORBACK* меняется все. При переходе по *ANDBACK* меняется все, кроме *ORBACK.*

При назначении вычислительного элемента для участия в процессе вывода ему назначается предикат тела с 4 вершинами, описанными выше.

Для одноименных переменных используется разделение структур данных, собственно переменная указывается один раз в утверждении.

Различают следующие множества:

1) множество *PT* всех предикатов;
2) множества *CT* описаний клозов с одинаковыми предикатами заголовка;
3) множества *HT* аргументов предикатов заголовков клозов;
4) множества *BT* предикатов тел клозов;
5) множества *AT* аргументов предикатов тела клозов.

Элемент d множества аргументов предиката заголовка AT или предиката тела BT является связанным, если имеется среди аргументов предиката тела или предиката заголовка элемент, изменяющийся точно так же как и d. При рассмотрении таких элементов устанавливается признак дереференсирования.

В случае связанного, один аргумент является элементом нескольких множеств. Выражение – является элементом (аргументом) некоторого множества – тождественно выражению – является связанным с некоторым элементом (аргументом) некоторого множества. Аргумент функтора также является множеством.

Для реализации алгоритма логического вывода с элементами распараллеливания выделены операции:

- поиск элемента множества *BT* предикатов тела клоза;
- поиск элемента множества *PT* всех предикатов
- поиск элемента множества *CT* клозов;
- поиск аргумента предиката заголовка (множества *HT*);
- сравнение аргументов предиката тела *AT* и предиката заголовка *HT*.

При переходе *ORBACK* возможны следующие случаи унификации аргументов:

1) аргумент предиката тела уже был унифицирован;
2) аргумент предиката тела не был унифицирован до вхождения в клоз для доказательства.

В первом случае, при возврате рассматривается аргумент, который был уже унифицирован до вхождения в предикат тела для доказательства, т.е. ссылка на него, во втором – сам элемент, который не был унифицирован до вхождения в предикат тела для доказательства.

Аппаратные средства реализации логического вывода. Структура средств реализации логического вывода изображена на рис. 3. Программа, написанная на декларативном языке (языке логического программирования типа Пролог), преобразуется в списковую структуру. Каждый из анализаторов логического вывода пытается выполнить унификацию запроса с одним из клозов с таким же именем предиката в заголовке, имеющимся в программе. Каждый из анализаторов логического вывода может получить свой клоз. После доказательства всех предикатов из тела запроса (получении пустого клоза в результате вывода) получаем результат истинности запроса.

При выводе возможна организация в памяти пулов, относящихся к выводу, выполняемому соответствующим анализатором вывода, с целью получения независимых областей памяти, относящихся к данному анализатору.

Задача планирования параллельного логического вывода сводится к решению следующих подзадач:

1) определение блоков данных (пулов) для выполнения одним вычислительным элементом (ВЭ);
2) выбор ВЭ для исполнения пула;
3) определение условия начала исполнения пула.

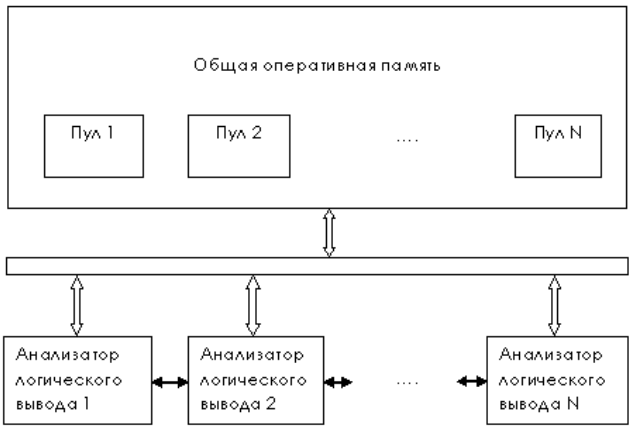

Рис. 3. Аппаратные средства реализации логического вывода

Для их решения используется как статическое планирование (до начала доказательства запроса), так и динамическое (во время доказательства). Необходимость последнего вызвана невозможностью определения до начала выполнения логического вывода.

Статическое планирование осуществляется при трансляции с декларативного языка во внутренние структуры данных. Первым начинает доказательство запроса ВЭ с первым сформированным пулом. Последующие ВЭ начинают работать по мере передачи им результатов унификации с ВЭ, приступивших к работе ранее.

Динамическое планирование параллельного логического вывода осуществляется диспетчером на универсальном процессоре (UCPU) и диспетчерами на однородных вычислительных элементах (ВЭ). ВЭ соединен с UCPU и другими ВЭ общей шиной. Приоритет захвата динамический и соответствует приоритету назначений на ВЭ. Пул назначенный ранее придает ВЭ более высокий приоритет. Наибольший приоритет закреплен на UCPU. Для доступа к своей памяти UCPU и ВЭ используют свои внутренние шины. При выборе альтернатив используется их расположение в памяти в порядке возрастания адресов. При этом не является обязательным их следование непосредственно друг за другом. Элементы альтернативы также располагаются последовательно друг за другом.

Разработанное представление декларативного описания и алгоритм доказательства запроса позволили сформулировать требования к структурам представления данных для параллельного логического вывода, уменьшить число шагов унификации для двух аргументов. На основании алгоритмов динамического планирования определены основные функции для обеспечения протокола интерфейса. Алгоритмы статического и динамического планирования позволяют выделять области кода для независимого исполнения, что дает возможность параллельного исполнения различных участков декларативного описания, по объему превышающей память одного ВЭ.

Заключение. В результате проведенного анализа выделены основные причины необходимости разработки описания архитектуры микропроцессорных систем. Предложен подход к построению описания, в котором, в отличие, от обычно используемых средств, вводится описание процессов операционной системы, средств трансляции. Подход базируется на технологии HAL/S. Показана связь описания процессов с декларативным языком, приведены основные элементы языка описания зависимостей процессов ОС. Для поддержки функционирования описания предложена обобщенная архитектура АРМ моделирования процессов микропроцессорной системы, приведена возможная архитектура классов реализации декларативного языка. Для обеспечения построения описания процессов на языке (R, M, A) уже имеющихся приложений в структуру АРМ введены средства декомпиляции. С целью повышения быстродействия системы моделирования на базе описания предлагаются модель параллельного выполнения логического программирования, а также соответствующая аппаратурная поддержка. При выполнении на параллельной аппаратурной реализации возможно линейное возрастание производительности с увеличением числа используемых ВЭ.

СПИСОК ЦИТИРОВАННЫХ ИСТОЧНИКОВ

1. Бибило, П.Н. Основы языка VHDL. – Мн.: Солон-Р, 2000.
2. Майерс, Г. Архитектура современных ЭВМ; в 2-х книгах. – М.: Мир, 1985.
3. Terence W. Pratt, George D. Maydwell Experience with formal semantic definition of HAL/S Symposium on Compiler Construction Proceedings of the 1982 SIGPLAN symposium on Compiler construction. Boston, Massachusetts, USA P: 327 – 333, 1982 [Электронный ресурс]. Режим доступа: http://ntrs.nasa.gov/archive/ nasa/casi.ntrs.nasa.gov/19890069053_1989069053.pdf
4. Stephen H. Edwards Logic Programming in C++. OOPSLA'02 November 4-8, 2002, Seattle, Washington, USA [Электронный ресурс]. Режим доступа: http://courses.cs.vt.edu/~cs5314/Spring02/Edwards-LP.pdf

Материал поступил в редакцию 25.11.10

KHVEDTCHUK V.I. The approach to the description of architecture of microprocessors

The problem of designing of the computer and in modern conditions is actual, not looking on development of architecture Intel, etc. Overcoming break between the equipment and program maintenance is still necessary.

The approach to the description of architecture of the computer is offered. As a result of lead analyses are selected the primary goals of the description of architecture of the computer. The approach to the description of architecture of microprocessor system is offered. In difference, from usually used means, the description of processes of operational system, means of translation is entered. Means HAL/S as analogue of the approach for construction of the environment of the description of processes in architecture of the computer are selected. Communication of the description of processes with declarative language is shown, basic elements of language are resulted. Purpose of the last is the description of dependences of processes of OS. Are offered architecture of an automated workplace of modelling of microprocessor system, the possible architecture of classes of realization of declarative language is resulted. The model of parallel performance of logic programming is resulted. At performance on parallel hardware realization probably linear increase of productivity with increase in number used computing elements.

УДК 004.514.62

Костюк Д.А., Костюк К.Л., Тавониус К.А.

СРЕДСТВА ИНТЕРНЕТ-НАВИГАЦИИ НА ОСНОВЕ МАСШТАБНЫХ ПРЕОБРАЗОВАНИЙ

Введение. В последнее время увеличивается наглядность элементов интерфейса, расположенных на периферии центральной (рабочей) части окна компьютерного приложения. В частности, набирает популярность использование уменьшенного масштаба изображений, не находящихся в фокусе работы пользователя – не в последнюю очередь в связи с ростом разрешающей способности дисплеев, делающей уменьшенные изображения объектов более информативными [1].

В средствах Интернет-навигации на сегодняшний день масштабные преобразования представлены в основном в экспериментальных интерфейсах, использующих миниатюры веб-страниц в качестве ярлыков. Однако широкополосный доступ и новые веб-технологии (включая перевод ряда пользовательских приложений в форму интернет-сервисов) изменяют модель взаимодействия с браузером, и в первую очередь значительно увеличивают число постоянно открытых веб-страниц. Разработчики интерфейсов ищут решения, наиболее подходящие к изменившимся условиям; в русле данных изысканий находятся и попытки увеличить наглядность навигации с помощью миниатюр.

В данной работе нами представлено развитие подхода к масштабированию веб-страниц на основе аналогии периферического зрения. Периферическое зрение (ПЗ), называемое часто также боковым или палочковым зрением, играет важную роль в ориентировании человека в окружающей среде [2, 3]. Фоторецепторы сетчатки делятся на два типа – колбочки и палочки. Основная масса колбочек сосредоточена в центральной части сетчатки, называемой желтым пятном. Центральное (колбочковое) зрение дает возможность рассматривать мелкие детали и опознавать предметы, а периферическое (палочковое) служит в основном для ориентирования в пространстве, обнаружения предметов и восприятия различных движений.

При работе со статическими документами палочковое зрение играет вспомогательную роль, облегчая запоминание расположения документов и их элементов. Однако многие современные интернет-ресурсы, в особенности построенные по технологии web 2.0, представляют собой динамически изменяющиеся объекты, самостоятельно обновляющие свой контент без полной перезагрузки страницы. В подобных системах роль ПЗ возрастает.

Нормальные границы поля зрения для одного глаза составляют [2]: по горизонтали: к виску – 90–100°, к носу – 50–60° (всего 140–160°); по вертикали: вверх – 50–60°, вниз 60–75° (всего 110–135°). Поле, одновременно охватываемое двумя глазами, по горизонтали несколько больше 180°, а по вертикали – около 120°. При вращении глаз наибольшее отклонение зрительных осей составляет ±45–50°. На рис. 1 показаны горизонтальные и вертикальные углы поля зрения (сектора ПЗ отмечены 2 и 3), а также характерные углы при правильной посадке оператора перед компьютерным дисплеем.

Из-за ощущаемой на интуитивном уровне естественности восприятия [3], связь с аналогией ПЗ присутствует в интерфейсах ряда наиболее популярных программных продуктов, вне зависимости от того, являлся ли соответствующий выбор дизайнера сознательным или также интуитивным.

Далее мы рассмотрим несколько подходов к организации работы с веб-страницами в контексте их связи с аналогией ПЗ и влиянием данной связи на эргономику интерфейса.

1. Использование периферийных областей экрана в веб-браузере. На сегодняшний день стандартом для программ веб-серфинга является механизм табов, совмещающий в одном окне несколько загруженных страниц и позволяющий переключать отображаемую страницу выбором соответствующей закладки — таба — в полосе, расположенной в верхней части окна. По существу данный механизм является аналогом панели задач. Веб-браузеры с табами пришли на смену прежней популярной модели однооконного интерфейса, в которой каждому окну с загруженной веб-страницей соответствовала кнопка на общесистемной панели задач или ее аналоге. Быстрое принятие табов пользователями можно объяснить следующими факторами:

Костюк Дмитрий Александрович, *к.т.н., доцент кафедры ЭВМ и системы Брестского государственного технического университета.*
Тавониус Кирилл Андреевич*, студент факультета электронно-информационных систем Брестского государственного технического университета.*
Беларусь, БрГТУ, 224017, г. Брест, ул. Московская, 267.
Костюк Кирима Львовна*, к.м.н., врач ЛПУ № 2.*