Раздел VI. Вычислительные комплексы нового поколения и нейрокомпьютеры

УДК 621.3.049.771.14

В.С. Гаврилов, Д.И. Рыжова, А.Н. Щелоков

МЕТОД УСКОРЕННОГО МОДЕЛИРОВАНИЯ ГЕТЕРОГЕННЫХ МНОГОЯДЕРНЫХ СИСТЕМ НА КРИСТАЛЛЕ С РЕКОНФИГУРИРУЕМОЙ ПАМЯТЬЮ

Целью данной работы является разработка метода и алгоритма для достижения разумного компромисса между скоростью и полнотой моделирования в маршруте совместной программно-аппаратной верификации систем на кристалле (СнК) «Мультикор». Отличительной особенностью предложенного подхода является конвейерная обработка инструкций при моделировании отдельных устройств с сохранением конкурентного доступа отдельных компонентов к ресурсам модели, а также моделирование гетерогенных многоядерных СнК с реконфигурируемой памятью обеспечивающее решение проблемы синхронизации многоядерных систем при существенном увеличении производительности по сравнению с моделированием RTL модели.

Моделирование систем на кристалле; синхронизация многоядерных систем; гетерогенные многоядерные СнК с реконфигурируемой памятью.

V.S. Gavrilov, D.I. Ryzhova, A.N. Schelokov

THE METHOD OF ACCELERATED SIMULATION OF HETEROGENEOUS MULTICORE SYSTEMS ON CHIP WITH RECONFIGURABLE MEMORY

The aim of this work is the development methods and algorithms is to achieve a reasonable compromise between speed and completeness of modeling in the route of hardware-software verification systems on chip (SoC) «Multicore». A distinctive feature of the proposed approach is the pipeline processing of instructions in the simulation of separate devices with saving concurrent access of the individual components to the resources of the model, and the heterogeneous multicore SoC with reconfigurable memory modeling, providing synchronization solutions for multicore systems with a significant increase in performance compared with the simulation of RTL models.

Systems on chip modeling; multicore synchronization; heterogeneous multicore SoC with reconfigurable memory.

Введение. Развитие микроэлектроники позволило создавать и широко использовать микросхемы, оснащенные программируемыми микропроцессорами и банками ПЗУ и ОЗУ. Отладка программного обеспечения (ПО) является неотъемлемой частью процесса разработки таких устройств и занимает около 50 % временных затрат. Отладка сложных прикладных программ на реальной аппаратуре затруднена. В связи с этим моделирование является одним из основных и притом доступных средств отладки прикладного ПО, а также разработки системы тестов для будущего кристалла.

Наиболее популярным подходом является моделирование на базе языков описания аппаратуры Verilog [2] или VHDL [3]. Использование этих языков позволяет построить полную достоверную модель, отладить (верифицировать) архи-

тектуру и систему команд, что, в конечном счете, позволяет спроектировать систему на кристалле (СнК). Однако, ввиду сложности полного моделирования очереди событий, использование модели Verilog на этапе программно-аппаратной верификации требует больших вычислительных затрат.

Противоположным подходом является эмуляция многоядерных систем в системе команд моделирующего компьютера. К числу таких средств моделирования СнК относится система Open Virtual Platforms (OVP) [4]. OVP обеспечивает динамическую трансляцию команд в двоичный код моделирующего компьютера. Основными преимуществами OVP являются высокая скорость моделирования и открытые функции API, упрощающие разработку сложных моделей.

Несмотря на все очевидные преимущества, OVP является в первую очередь транслятором исполняемого кода для одного процессора. Это накладывает серьезные ограничения на моделирование многопроцессорных систем. Кроме того, периферийные устройства реализованы в виде пассивных регистровых блоков, что существенно ограничивает моделирование. Основной причиной такой реализации является невозможность быстрой синхронизации нескольких элементов моделируемой микросхемы. При переключении между элементами весь транслированный код одного элемента (например, процессора) подменяется кодом для другого. В целях повышения производительности, данные операции выполняются не чаще 100 тысяч инструкций, что негативно сказывается на межпроцессорной синхронизации.

Методы моделирования SIM3x. Для достижения разумного компромисса между скоростью работы программы моделирования и точности результатов с учетом синхронизации микропроцессоров в ГУП НПЦ «Элвис» разрабатывается система SIM3x для моделирования СнК «Мультикор». Все устройства (процессоры, контроллеры прямого доступа в память, порты ввода-вывода и др.) работают независимо и на своей частоте.

Работа по выполнению инструкции отдельно взятого устройства может быть описана в функциональной форме $Y_i(t+\Delta_i) = F_i(X_{i,1}(t),...,X_{i,n}(t))$, где $X_{i,j}, j = 1,...n$ – входные данные (на регистрах или в памяти), Y_i – выходные данные, t – текущее время моделирования, F_i – выполняемая функция, Δ_i – задержка или время выполнения инструкции. Через $A(X_{i,j}), A(Y_i)$ будем обозначать адреса входных и выходных данных соответственно. Последовательное моделирование отдельных блоков должно учитывать конвейерную обработку инструкций на фиксированном наборе стадий конвейера, таких как чтение команды (RC), декодирование команды (DC), чтение входных данных $X_{i,j}, j = 1,...n$ (ER), непосредственно выполнение команды (E1, E2, …), запись выходных данных Y_i (EW). Для примера в табл. 1 изображена последовательность выполнения команд 7-истадийного конвейера.

Если же рассматривать моделируемые устройства на более высоком уровне, то следует говорить об очереди событий с последовательным моделированием отдельных устройств во времени. Для этого работа каждого устройства разбивается на малые операции – шаги (табл. 1), «время» исполнения, которого соответствует частоте устройства. Пусть $i = I_u(j,k);\ \ j = 1,..., j_{\max};\ \ k = 0,..., k_{\max}$ – номер инструкции устройства с порядковым номером *u*, выполняемой на *j*-й стадии конвейера в момент времени $t = k \cdot P_u = \dfrac{k}{f_u}$, где P_u, f_u – соответственно период и частота устройства с порядковым номером *u*. Тогда $I_u(j,k) = I_u(j-1,k-1)$ для

всех $j = 2,..., j_{\max}$; $k = 1,..., k_{\max}$. В идеальном случае полной загрузки конвейера $I_u(j,k+1) = I_u(j,k)+1$. Обработка инструкций в рамках *k*-го такта отдельно взятого устройства сводится к последовательной обработке инструкций столбца матрицы $I_u(j,k)$ в обратном порядке снизу вверх: $j = j_{\max}, j_{\max}-1,..,1$. Обратный порядок обеспечивает контроль блокировки операции чтения данных (ER) в ситуации с задержкой записи данных (EW) на этапе выполнения предыдущей операции при использовании общих регистров или памяти (в табл. 1 – задержка выполнения операции 5, вызванная записью в операции 2).

Таблица 1

Последовательность выполнения команд 7-и-стадийного конвейера

№	Код стадии	k=00	k=01	k=02	k=03	k=04	k=05	k=06	k=07	k=08	k=09	k=10	k=11	k=12
1	RC1	1	2	3	4	5			6	7	8	9	10	11
2	RC2		1	2	3	4	5			6	7	8	9	10
3	DC			1	2	3	4	5			6	7	8	9
4	ER				1	2	3	4			5	6	7	8
5	E1					1	2	3	4			5	6	7
6	E2						1	2	3	4			5	6
7	EW							1	2	3	4			5

Для реализации очереди событий на межпроцессорном уровне предлагается использование двух связных списков устройств. Элементом такого списка является упорядоченная тройка (t,u,p), где *t* – время выполнения, *u* – номер устройства, *p* – приоритет устройства для разрешения конфликтов с близкими или равными значениями *t*. Первый список $L_1 = \{e_l\} = \{(t_l, u_l, p_l), l = 1,..., l_{1\max}\}$ – список активных устройств, второй список $L_2 = (f_l\} = \{(t_l, u_l, p_l),\ l = 1,..., l_{2\max}\}$ – список неактивных или замороженных устройств. Алгоритм выполнения очередного шага модели включает следующие операции:

1) выбрать наиболее готовое устройство $e_1 = (t_1, u, p)$;
2) выполнить шаг наиболее готового устройства в соответствии со списком операций столбца матрицы $I_u(j,k);\ \ j = j_{\max}, j_{\max}-1,..,1;\ \ k = \frac{t_1}{P_u}$;
3) при обнаружении блокировки перевести e_1 из L_1 в L_2;
4) при отсутствии блокировки изменить системное время исполнения данного устройства: $t_1^* = t_1 + \Delta_i$, переместить $e_1^* = (t_1^*, u, p)$ в списке L_1 в соответствии с требованиями порядка

$$\forall i, j : (i < j) \Rightarrow t_i < t_j \vee (t_i = t_j \ \&\ p_i > p_j).$$

По мере готовности входных данных $X_{i,1}(t),..., X_{i,n}(t)$ выполняется разблокировка замороженных устройств.

Механизм обработки гетерогенных многоядерных СнК с реконфигурируемой памятью. Платы «Мультикор» содержат большое число компонентов. Каждый компонент, за исключением центрального процессора, настраивается посредством адресуемых регистров.

Согласно проведенному профилированию, 80 % операций, производимых в прежней программной модели «Мультикор» являлись операциями чтения или записи регистров (проще говоря, синхронизацией значений регистров с памятью). Другой проблемой является то, что один и тот же регистр в составе разных плат «Мультикор»

может иметь не только разные маски чтения или записи, но и разные адреса $A(X_{i,j})$, а также разные начальные значения. По результатам профилирования было принято решение о переориентации функции доступа к памяти на регистры, в результате чего скорость моделирования прежней программной модели была удвоена.

В реальной аппаратуре для синхронизации регистров используется сигнальный механизм доступа к данным регистров. В разрабатываемой модели предлагается использовать похожий принцип. Каждое устройство производит поиск регистра в общей базе регистров, после чего добавляет реакцию на определенный доступ к регистру.

Предлагается при доступе к памяти проверить адрес памяти в таблице соответствия регистров выделенным ячейкам памяти – $T(A)$, где $A = A(X)$ − адрес чтения для $(X = X_{i,j})$ или записи при $(X = Y_i)$. Если адрес найден $(T(A) \neq NULL)$, то производится либо доступ к регистру, либо останов моделирования (отладочная точка останова). Таблица может быть организована в виде сбалансированного двоичного дерева. Не смотря на это, постоянное обращение к ней остается достаточно дорогой операцией. Имеется возможность снизить количество обращений к таблице до минимума. Для этого будем использовать специальный флаг, а именно специальное значение – метку λ – для тех ячеек памяти, которые в реальности соответствуют регистрам или точкам останова. Если при доступе к ячейке ее текущее значение равно значению-метке $(X_{i,j} = \lambda)$, то в этом и только в этом случае следует производить поиск в таблице $T(A(X_{i,j}))$. Каждая ячейка памяти содержит 32 бит (2^{32} возможных значений), тем самым использование метки снижает количество обращений к таблице соответствия более чем в 10^9 раз.

Экспериментальные результаты. Предложенные методы и алгоритмы были реализованы на языке C++. В качестве испытаний были рассмотрены специальные тесты производительности микропроцессоров: coremark, Dhrystone, whetstone, linpack. Производилось сравнение Verilog-модели (RTL)[2], ARM RTSM[5], OVR [4] и предлагаемой в рамках данной работы модели SIM3.

Результаты представлены в табл. 2. Малая скорость Verilog вызвана, в первую очередь, значительными производительными затратами на точность моделирования, необходимую для имитации задержек передачи данных, потоков данных (DataFlow), очередей событий. Наиболее быстрой является платформа OVP. Подобная скорость вызвана в первую очередь применяемым подходом: OVP использует бинарную трансляцию исходных кодов для моделируемого процессора в коды для персонального компьютера. Однако структура OVP накладывает существенные ограничения на моделирование многоядерных систем.

Таблица 2

Экспериментальные результаты испытаний

Тест	Скорость работы алгоритма (инструкций / сек.)			
	RTL	ARM RTSM	OVP	SIM3x
coremark	79к	2120к	93000к	8400к
dhrystone	81к	2134к	95500к	8300к
whetstone	78к	1800к	14950к	3800к
linpack	73к	1850к	13500к	4280к

Платформа SIM3x, основанная на предложенных методах, обеспечивает значительное ускорение по сравнению с платформой Verilog, и, в тоже время, поддерживает механизм синхронизации при использовании общих регистров или памяти. Это обеспечивает разумное сочетание точности моделирования и времени, необходимого для моделирования сложных прикладных программ.

Заключение. В рамках работы проанализированы различные подходы к моделированию СнК типа «Мультикор». Показано, что наиболее полным и точным (но медленным) является моделирование RTL на основе Verilog. Альтернативный вариант моделирования на основе платформы OVP обеспечивает наибольшую скорость, однако не гарантирует синхронизацию отдельных процессоров в многоядерной системе.

Для достижения разумного компромисса между скоростью и полнотой моделирования разработана система моделирования SIM3x. Отличительными особенностями системы является конвейерная обработка инструкций при моделировании отдельных устройств с сохранением конкурентного доступа отдельных компонентов к ресурсам модели. Использование специальной таблицы соответствия регистров ячейкам памяти существенно повышает скорость механизма синхронизации между процессорами многоядерной системы.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Солохина Т.В.* Архитектура DSP-акселераторов на базе платформы "Мультикор" для суперкомпьютеров нового поколения // Проблемы разработки перспективных микро- и наноэлектронных систем – 2008. Сборник научных трудов / Под общ. ред. А.Л. Стемпковского. – М.:ИППМ РАН, 2008. – С. 415-418.
2. *Samir Palnitkar*. Verilog HDL. A Guide to Digital Design and Synthesis // Pearson Education, ISBN: 81-7758-918-0.
3. *Бибило П.Н.* Основы языка VHDL. – М.: Солон-Р, 2002.
4. Overview. First time use of OVP simulators and emulators with MIPS processors and platforms // http://www.ovpworld.org/first_MIPS.php.
5. Emulation Baseboard Real-Time System Model User Guide // http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.dui0424b/index.html.

Статью рекомендовал к опубликованию д.т.н., профессор А.Л. Глебов.

Гаврилов Виталий Сергеевич – Федеральное государственное автономное образовательное учреждение высшего профессионального образования «Национальный исследовательский университет «МИЭТ»; e-mail: vitaly.s.gavrilov@gmail.com; 124498, Москва, Зеленоград, проезд 4806, 5; тел.: 84997381388; кафедра проектирования и конструирования интегральных микросхем (ПКИМС); аспирант.

Рыжова Дарья Игоревна – Федеральное государственное бюджетное учреждение науки Институт проблем проектирования в микроэлектронике Российской академии наук (ИППМ РАН); e-mail: ryzhova_d@ippm.ru; 124365, Москва, Зеленоград, ул. Советская, 3; тел.: 84997299890; отдел автоматизации проектирования цифровых схем; инженер-исследователь.

Щелоков Альберт Николаевич – e-mail: schan@ippm.ru; тел.: 84997299845; зам. директора.

Gavrilov Vitaly Sergeevich – National Research University of Electronic Technology; e-mail: vitaly.s.gavrilov@gmail.com; 5, passage 4806, Moscow, Zelenograd, 124365, Russia; phone: +74997381388; the department of IC design and engineering; postgraduate student.

Ryzhova Daria Igorevna – Institute for Design Problems in Microelectronics of the Russian Academy of Science (IPPM RAS); e-mail: ryzhova_d@ippm.ru; 3, Sovetskaya street, Moscow, Zelenograd, 124365, Russia; phone: +74997299890; the department of digital design automation; research engineer.

Schelokov Albert Nikolaevich – e-mail: schan@ippm.ru; phone: +74997299845; deputy director.