УДК 004

Системный уровень проектирования IP-блока DSP процессора семейства NeuroMatrix

В.В. Дементьев

ЗАО НТЦ «Модуль», v.dementev@module.ru

***Аннотация* — В докладе описываются возможные применения высокоуровневых поведенческих моделей на системном уровне проектирования (ESL). Также обосновываются требования к программным моделям, учитывающим особенности IP-блока NeuroMatrix Processing Unit (NMPU). IP-блок содержит 32-разрядный управляющий RISC-процессор, матрично-векторные вычислительные сопроцессоры для цифровой обработки данных (DSP), блоки внутренней памяти и системный интегратор для связи компонент системы.**

***Ключевые слова* — функциональная модель, потактовая модель, конвейер.**

I. Введение

Высокая степень интеграции современных СБИС позволяет размещать на кристалле сложные вычислительные комплексы [1, 2]. При проектировании таких СБИС возникают проблемы, связанные с необходимостью оценки показателей системы на ранних стадиях проектирования и системной верификации, т.е. проверки корректного взаимодействия узлов системы, которая должна производиться как можно раньше [3].

Для сокращения времени проектирования происходит переход от разработки специализированных СБИС к построению так называемых систем на кристалле, допускающих повторное использование уже хорошо отлаженных сложно-функциональных блоков, в том числе от сторонних производителей.

Наличие в системе программируемых блоков (процессорных ядер) приводит к тому, что часть алгоритмов, для решения которых разрабатывается система, реализуется в виде встроенного программного обеспечения. Становится необходимой параллельная разработка программного и аппаратного обеспечения. Также необходимо проводить совместную отладку и верификацию программно-аппаратного комплекса.

Требования ранней верификации проекта диктуют необходимость применения высокоуровневых поведенческих моделей на системном уровне проектирования. Под системным уровнем подразумевается все, что лежит выше уровня RTL. Любая модель на уровне ESL будет называться программной.

В докладе описывается предполагаемый подход к выполнению системного уровня проектирования процессорных IP-блоков NMPU и устройств на их основе. ESL базируется на применении программных моделей, которые полезны одновременно разработчикам программного и аппаратного обеспечения.

Программные модели, применяемые на системном уровне, должны позволять:

1) отлаживать программное обеспечение;
2) оценивать производительность;
3) проверять эффективность новых архитектурных решений;
4) допускать повторное применение.

II. Системный уровень – ESL

Рост сложности разрабатываемых устройств обуславливает появление системного уровня проектирования и верификации [3, 4]. На уровне ESL происходит проверка возможности реализации требуемых алгоритмов, разделение на аппаратные и программные блоки, определение количественных показателей. Место уровня ESL в обобщенном маршруте проектирование показано на рис. 1.

Рис. 1. Стадии проектирования

Системный уровень нуждается в создании поведенческих моделей на высокоуровневых языках программирования, которые должны позволять быстро создавать прототип проектируемой системы, обеспечить возможность легкого внесения изменений и быть использованы повторно. В дальнейшем подобные модели могут применяться для решения задач:

1) функционального моделирования всей системы, проверки реализуемости алгоритмов и определения основных составляющих SoC;

2) поведенческого моделирования, анализа архитектуры и проверки новых архитектурных решений, предварительного определения основных количественных показателей;

3) отладки программного обеспечения на ранних стадиях проектирования (до появления RTL-модели или прототипа на FPGA);

4) системной верификации в качестве эталонной модели, создания системы для сквозного тестирования на всех уровнях абстракции;

5) функциональной (автономной) верификации отдельных блоков, высокоуровневого синтеза в RTL.

Для задач верификации в модели должен быть поддержан интерфейс подключения к тестовому окружению уровня RTL-модели. На этом уровне тестовое окружение часто проектируется на SystemVerilog с применением методологии UVM [5, 6]. Благодаря интерфейсу DPI возможно подключить модели, написанные на языках общего назначения, например на C++. Интерфейс поддерживается как со стороны тестового окружения, так и модели, за счет экспортируемых и импортируемых функций (переданных и возвращаемых параметров).

Появляется новый уровень абстракции – TLM (transaction level modeling), он нацелен на минимизацию проблем, связанных с уровнем RTL (низкая скорость моделирования и сложность функциональной верификации). Здесь появляется возможность проектировать модели блоков вычислительной системы, которые подходят как для отладки программ, так и для дальнейшей разработки аппаратуры. В конечном счете TLM-IP заменят RTL-IP [7]. Крупнейшие производители CAD предоставляют средства для проектирования программных виртуальных прототипов на системном уровне:

1) Virtual System Platform (Cadence), которая совместима с ускорителем эмуляции Palladium;

2) Virtualizer (Synopsys);

3) Vista Architect (Mentor Graphics).

Эти средства позволяют разрабатывать программное обеспечение, проводить системную верификацию, системный анализ и оптимизацию до появления аппаратной микроархитектуры (RTL/FPGA prototype). Они позволяют полностью контролировать проект. Программисты имеют полный набор средств для отладки программ. А разработчикам аппаратуры позволено наблюдать различные сигналы и оценивать различные характеристики системы (например, производительность). В своем составе средства виртуального прототипирования содержат набор моделей IP-блоков и процессорных ядер, написанных на SystemC/C++ согласно стандарту OSCI TLM 2.0. Таким образом, для совместимости с САПР программные модели должны поддерживать интерфейс, подходящий для уровня TLM, и, следовательно, должны быть написаны на C++/SystemC, который становится стандартом (IEEE 1666-2011) для системного уровня проектирования. С другой стороны, SystemC – библиотека языка C++, и, следовательно, модель может разрабатываться на любом компьютере без спецсредств САПР, для которого есть компилятор (например, GCC),.

Начиная разработку моделей, необходимо определить интерфейс и нужный уровень детализации (определяет точность и производительность). Основным фактором, определяющим детализацию модели, является уровень абстракции, для которого пишется модель, а также её назначение. На выбор интерфейса модели влияют средства проектирования/управления и окружение, с которым взаимодействует модель (среда исполнения).

Базой для построения точной модели (потактовой) должны служить более простые модели (функциональные), которые уже отлажены на ранних стадиях проектирования. При этом они должны иметь единый интерфейс. Это дает возможность использовать уже наработанное тестовое покрытие и минимизировать время построения моделей.

III. Структура NMPU

В этом и V разделе рассматриваются особенности систем с архитектурой NeuroMatrix, влияющие на структуру программной модели. NeuroMatrix отличается высокой степенью интеграции, функциональной сложностью и высокой производительностью для задач цифровой обработки данных. Обобщенная структурная схема IP-блока NMPU представлена на рис. 2. Более подробную информацию о структуре вычислительных систем на базе NeuroMatrix, их применении и программном интерфейсе (наборе команд и регистровом файле) можно получить из [8, 9]. В настоящее время разрабатывается новое поколение устройств (существуют тестовые образцы на кристалле), основанное на процессорном ядре следующего поколения NMC4.

В состав NMPU входят управляющий RISC-процессор (реализует конвейер) с матрично-векторными сопроцессорами VU/FPU (функциональные устройства – исполнительные стадии конвейера). Для снижения времени ожидания команд введен блок предвыборки IFU. Доступ к памяти и периферийным устройствам PU осуществляет системный интегратор SI, который транслирует запросы ядра или блока предвыборки в соответствующие блоки памяти, а также возвращает ядру запрошенные данные/команды. Память реализована в виде нескольких внутренних блоков SRAM (IMUx), также поддержан интерфейс к внешней динамической памяти. Для обеспечения нужной производительности ядро NMC4 обменивается данными с помощью развитой шинной структуры (SDATA/VDATA<n>), позволяя осуществлять одновременно несколько операций ввода/вывода за один процессорный такт.

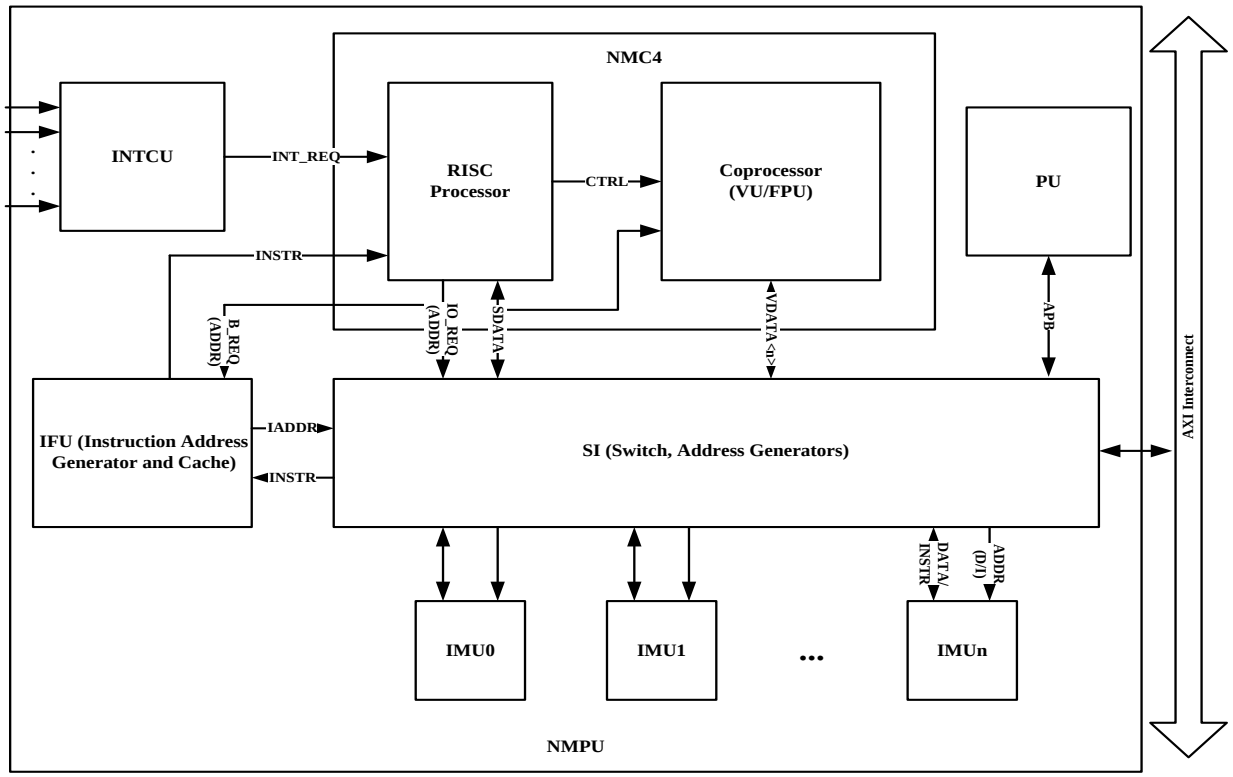

Рис. 2. Структурная схема NMPU

В отличие от предыдущих поколений у NMC4 появился сопроцессор для обработки данных в формате с плавающей точкой (IEEE754).

В различных реализациях системы может варьироваться состав и количество сопроцессоров, количество блоков памяти и набор периферийных устройств (совместимых с AMBA Peripheral Bus). Для взаимодействия с внешним миром поддерживается шинный интерфейс, совместимый со спецификацией AMBA AXI 3.0, а также развитая система прерываний (INTCU).

IV. Функциональная модель NMPU

Особенностью программной модели процессорной системы является то, что для неё не требуется специально разрабатывать тестовое окружение для генерации входных воздействий и сравнения выходных реакций с эталоном. Эти задачи берут на себя тестовые программы, которые исполняются на процессоре (минимальная единица воздействия это команда, т.е. тестовое окружение – сам процессор). Поскольку модель процессорной системы сама по себе является программой, для её запуска требуется интеграция с программным отладчиком (например, GDB) [10], основными задачами которого являются:

1) симуляция работы одного такта виртуальной процессорной системы (GDB stub);
2) размещение кода в памяти программ/данных (разбор файлов формата ELF);
3) контроль над выполнением целевой программы (точки останова);
4) просмотр содержимого памяти;
5) доступ к состояниям внутренних регистров;
6) дизассемблирование объектного кода.

Общая схема взаимодействия программного отладчика с моделью процессорной системы проиллюстрирована на рис. 3.

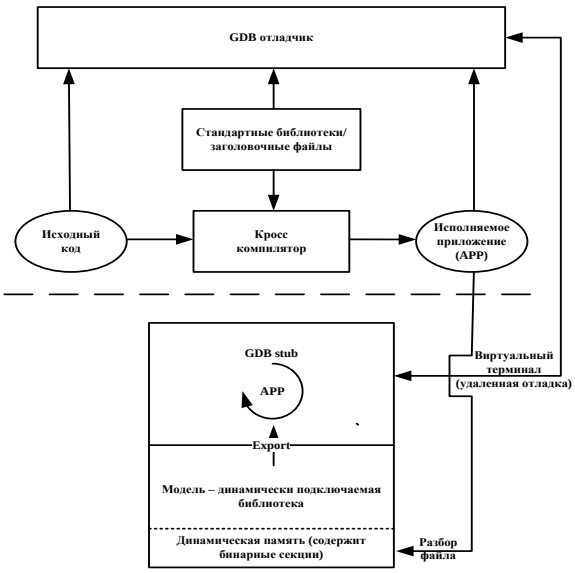

Рис. 3. Взаимодействие отладчика с моделью

Тестовые программы должны проверять функционирование всех узлов системы и включать в свой состав прикладные программы, для которых разрабатывалась система. Это снижает вероятность пропустить ошибки реализации.

Для процессорного ядра NMC4 разработано инструментальное программное обеспечение:

1) GNU компилятор;
2) GDB отладчик (совместим только с ISS);
3) IDE на базе Eclipse;
4) симулятор набора команд (ISS).

Симулятор набора команд NeuroMatrix представляет собой функциональную модель, на которой все инструкции выполняются за один такт строго последовательно (реализован только программный интерфейс). Такая модель обеспечивает корректное выполнение целевых инструкций, но не отображает правильные значения длительностей операции, т.е. не учитывает архитектурные особенности процессора (в частности, конвейер).

Система команд описана на языке PPDL (processor and peripheral description language) [11]. В настоящее время существует только прототип языка PPDL, который используют для выполнения заказных работ. Концепция языка PPDL позволяет автоматизировать получение декодера набора команд и его дизассемблера из единого описания функциональности каждой инструкции (транслятор генерирует код на языке C++). Это позволяет быстро разрабатывать функциональные модели и вносить изменения в набор команд. Язык поддерживает ряд абстракций, которые позволяют избежать трудностей, описанных в [12]. В частности, моделируется работа регистров (разные значения при чтении/записи до синхронизации) и очереди типа FIFO, что дает возможность корректно моделировать работу команды. Предоставлен интерфейс к отладчику (управляет работой ядра процессора) и гибкий интерфейс для подключения моделей пассивных устройств (шины, память), которые могут быть описаны на языке C++.

Однако язык PPDL имеет ряд недостатков. Полученная модель работает на основе интерпретации инструкций (для функциональной модели медленно) и поддерживает только один поток исполнения (моделируется одно ядро). Для повышения точности язык содержит методы блокировки доступа к регистрам на заданное число тактов. Разработка потактовой модели требует проводить тренинг [13] на точной RTL-модели или отладочной плате с применением большого числа тестовых программ. Это не гарантирует необходимую точность исполнения произвольной программы. Также нет возможности быстро модифицировать модель при внесении изменений в архитектуру системы (конвейер). Т.е. факторы, обуславливающие задержки, не детерминированы.

Ещё один недостаток – отсутствие интерфейса, позволяющего обмениваться данными с другими активными устройствами (нет элементов многопоточного программирования). Модели периферийных устройств можно разбить на две составные части: память (набор регистров – пассивная часть) и исполнительное устройство (конвейер – активная часть). С программной точки зрения взаимодействие с моделью ядра процессора обеспечивается через интерфейс памяти. Такая концепция годится только для получения функционального описания. Основная трудность заключается в обеспечении взаимодействия и синхронизации моделей активных устройств, которые могут работать на разной частоте и поддерживают асинхронные режимы работы (прерывания).

Функциональные модели хорошо подходят для проверки реализуемости алгоритма на данном наборе команд, но не годятся для количественной оценки показателей системы, проверки новых архитектурных решений на ранней стадии проектирования аппаратуры и задач верификации. В дальнейшем предполагается использовать язык PPDL для функционального описания инструкций, из которого компилятор генерирует декодер и дизассемблер инструкций. Это дает возможность использовать уже отлаженное описание набора команд на языке PPDL. Благодаря тому, что PPDL-описание транслируется в код на языке C++, проблемы, связанные с отсутствием подходящих средств разработки потактовой модели и интерфейса с активными устройствами, можно устранить с помощью реализации дополнительных структур данных и методов для их обработки, а также использовать готовые библиотеки языка SystemC для создания поведенческих моделей цифровых устройств. Такой подход позволяет использовать все возможности языка C++ и при этом сохранить интерфейс для подключения внешнего программного отладчика и интерфейс для подключения модели системы памяти.

Таким образом, программная модель процессорного ядра, реализующая алгоритм работы конвейера и интерфейс для взаимодействия с другими устройствами, в том числе и от сторонних производителей, разрабатывается на языке C++ с использованием элементов SystemC (ядра событийного моделирования).

V. Структура конвейера NMC4

Основной частью процессорной системы является ядро NMC4. Оно реализует вычислительный конвейер и выполняет управление системой. RISC-процессор выполняет:

1) декодирование команд, аппаратный контроль конфликтов по данным и детектирование запрещенного сочетания инструкций;
2) необходимые предварительные адресные вычисления, установку запроса на обмен с памятью и изменение порядка следования команд;
3) арифметические и логические операции над скалярными данными;
4) управление работой векторных сопроцессоров VPU/FPU, которые выполняют цифровою обработку данных.

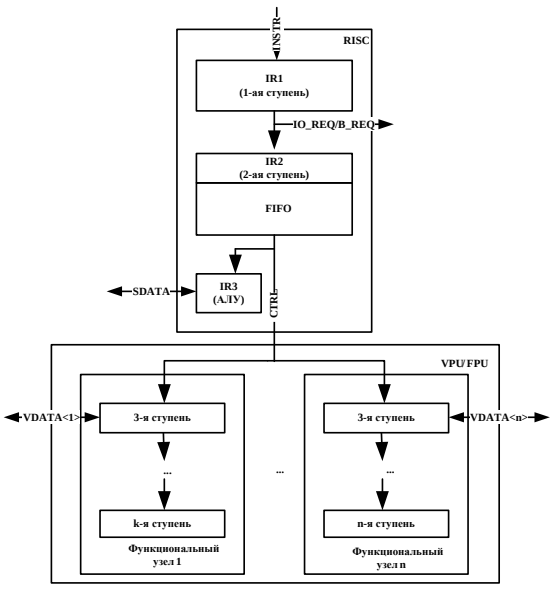

Рис. 4. Принципиальная схема конвейера

Принцип организации конвейера исполнения команд показан на рис. 4. Его основными особенностями являются:

1) наличие двух общих ступеней (первой и второй) для всех команд, причем на первой ступени осуществляется вычисление адреса данных для команды и модификация адресных регистров, а на второй организуется единая очередь команд, ожидающих своих данных перед выполнением;

2) несколько параллельных подконвейеров на третьей ступени (стадии выполнения операций), причём ввод и вывод данных осуществляется именно на данной ступени.

Также одной из особенностей работы конвейера является наличие отложенных команд, которые возникают при выполнении команд переходов. Это связанно с тем, что с момента установки запроса ветвления до фактического изменения порядка следования команд (нужная команда выбрана из памяти) проходит несколько тактов. Чтобы компенсировать простои в конвейере, введены отложенные команды, а блок предсказания ветвлений отсутствует.

Запрос на обмен с памятью для чтения/записи данных устанавливается на первой ступени конвейера IR1. Затем команда ожидает данные в очереди на второй ступени конвейера IR2. Когда требуемые данные готовы, команда переходит на третью исполнительную ступень конвейера IR3, на которой фактически происходит ввод/вывод данных, а также выполняются заданные вычисления. Задачи, связанные с выборкой команд из памяти, вынесены за пределы ядра.

За выборку команд отвечает блок предвыборки IFU. Он содержит адресный генератор, который устанавливает запросы на чтение очередной команды, а также буфер для хранения команд.

Доступ к памяти по запросам от ядра и блока предвыборки фактически устанавливает системный интегратор SI, который проводит арбитраж, сериализирует и коммутирует поступившие запросы. Также системный интегратор содержит адресные генераторы для блочного обмена данными без участия ядра.

Ввиду того, что чтение и запись происходят на одной ступени конвейера, имеется ряд особенностей. Чтение происходит естественным образом, а адрес и данные при записи устанавливаются на разных ступенях конвейера. Это реализуется с помощью специального механизма синхронизации подсистемы памяти.

Поскольку эффективность работы конвейера во многом определяет производительность архитектуры, необходимо иметь возможность моделировать его работу. Возможная производительность конвейера зависит от сочетания инструкций и расположения программ данных в памяти. Отсюда вытекает, что модель конвейерного исполнения команд должна объединять в себе модели ядра, системного интегратора, блока предвыборки команд и подсистемы памяти.

Структура программной модели повторяет структуру RTL-модели (уже разработанной системы) на уровне основных блоков: RISC, VU/FPU, SI, IMU. Это в дальнейшем позволит быстро модифицировать модель при изменении архитектуры и использовать её повторно. При этом обеспечивается корректное поведение отдельных блоков (интерфейсы упрощены), а не их фактическая реализация. Например, не нужно моделировать промежуточное представление инструкций (mop – микрооперации) на каждой ступени конвейера, достаточно обеспечить правильное протекание операций (их длительность) и конечный результат. Трудозатраты на разработку и дальнейшую поддержку программной модели гораздо ниже затрат на проектирование RTL-модели, а скорость моделирования на порядок выше.

VI. Модель конвейерного исполнения команд

В силу большой внутренней сложности процессорных систем для их разработки используются уже хорошо наработанные решения, следующее поколение семейства строится на основе предшествующих поколений. Таким образом, сохраняется программная совместимость устройств (система команд изменяется незначительно). При этом структура конвейера может изменяться сильно.

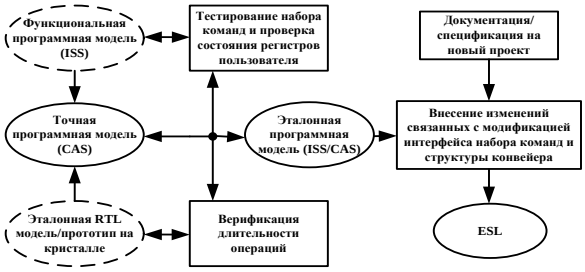

Рис. 5. Принцип повторного использования модели

Модель, позволяющая оценивать производительность конвейера процессора в зависимости от выполняемой программы (размещения в памяти, последовательности и сочетания инструкций), строится на основании анализа данных, полученных при моделировании поведения основных блоков системы на уже разработанном и отлаженном уровне RTL. Результаты такого анализа позволяют учесть архитектурные особенности при разработке модели (детерминировать причины возможных задержек). До появления прототипов или RTL-модели нет достаточной информации для построения точных моделей, в то время как для построения функциональной модели достаточно описания системы команд.

Благодаря преемственности поколений процессорных систем, программная модель текущего поколения является базисом для построения точных моделей следующих поколений, который после доработки в соответствии с техническими требованиями может применяться как эталон для дальнейшего проектирования. Идея повторного применения проиллюстрирована на рис. 5.

Блок-схема модели взаимодействия частей системы, построенная по результатам анализа поведения на уровне RTL, показана на рис. 6. Представлен упрощенный вариант модели, который отражает выполнение скалярных команд (без команд управления), выбираемых из внутренней памяти.

При выполнении операций доступа в память возможны конфликты. Они возникают, когда команды (IFU) и данные (RISC) выбираются из общей памяти. Такие конфликты разрешаются системным интегратором на основе приоритетов запросов. Конфликты по данным возникают, когда операции вычисления адреса и модификации адресных регистров (ступень IR1) используют те же ресурсы, что и арифметико-логические операции (ступень IR3). Конфликты по данным отслеживаются на аппаратном уровне (ступень IR0). Это достигается путем захвата требуемых ресурсов, которые освобождаются на ступени IR3 по завершении операции. При возникновении конфликтной ситуации по данным команда блокируется на ступени IR0. Долгие операции с памятью блокируют команды в общей очереди на ступени IR2 (дожидаются данных).

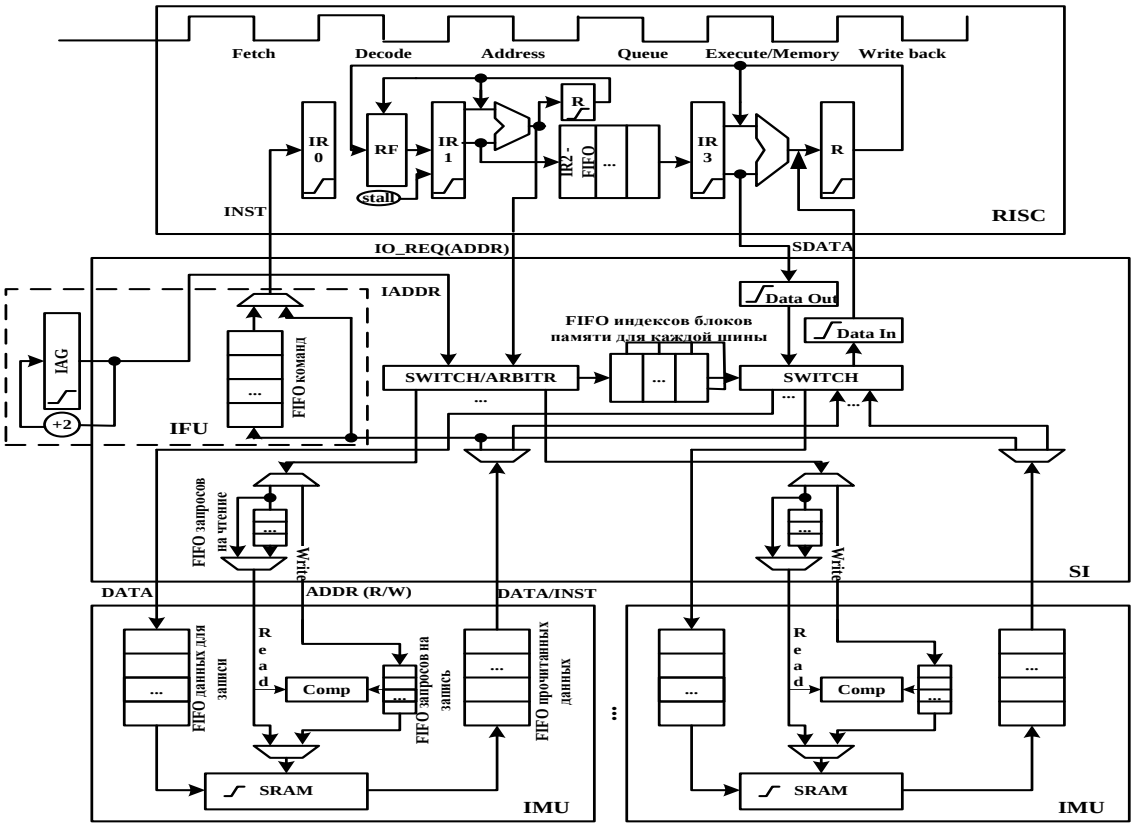

Рис. 6. Блок схема модели взаимодействия уровня транзакций

Такой детализации на уровне транзакций достаточно, чтобы моделировать возникновение конфликтных ситуаций и анализировать поведение конвейера при выполнении прикладных алгоритмов. Также допустимо вносить изменения для проверки новых аппаратных решений и оценки их эффективности.

Требования повышения производительности заставляют постоянно улучшать архитектуру систем. Нужная производительность может быть достигнута путём углубления конвейера. Однако при этом возникают проблемы, связанные с эффективностью его использования. Усугубляется влияние конфликтов по данным, вследствие чего появляются длительные простои конвейера. В работе [14] предлагается использо-

вать аналитический метод расчета для выявления оптимальной структуры конвейера с точки зрения производительности. Аналитические расчеты требуют экспериментального подтверждения. Программная модель позволяет быстро реализовать требуемые изменения.

VII. Заключение

Проведенные исследования показали целесообразность разработки и применения программных моделей для проектирования цифровых устройств, в частности, процессорных систем с конвейерным выполнением команд. При этом программные модели должны соответствовать следующим требованиям:

1) модель должна быть написана на C++/SystemC;

2) основываться на симуляторе инструкций;

3) исходные данные для разработки получены при проведении RTL-моделирования;

4) структура модели должна повторять строение RTL-модели;

5) точность должна быть достаточной для решения задач реального времени (оценка производительности).

Модель конвейерного исполнения команд позволяет оценить эффективность применяемых архитектурных решений и корректность разработанного программного обеспечения.

Литература

[1] Стешенко В., Руткевич А., Гладкова Е., Шишкин Г., Воронов Д. Проектирование СБИС типа “система на кристале”. Маршрут проектирования. Синтез схемы. Часть 1 // Электронные компоненты. 2009. № 1. С. 14–21.

[2] Евтушенко Н., Немудров В., Сырцов И. Методология проектирования систем на кристале. Основные принципы, методы, программные средства // Электроника: Наука, Технология, Бизнес. 2003. № 6. С. 7–11.

[3] Непомнящий О.В., Хныкин А.В. Анализ проектирования вычислительных систем на кристалле // Исследовательский наукоград. 2012. № 1. С. 42-46.

[4] Непомнящий О.В., Легалов А.И., Сиротина Н.Ю. Маршруты и технологии архитектурно-независимого проектирования при разработке сложных однокристальных схем специального назначения // Институт космических и информационных технологий СФУ. URL: http://ikit.sfu-kras.ru/files/ikit/8_Marshrut_proektirovania.pdf (дата обращения 31.03.2016).

[5] Путря Ф.М. Особенности использования возможностей объектно-ориентированного программирования SystemVerilog для функциональной верификации многоядерных СнК // Сб. трудов «Проблемы разработки перспективных микро- и наноэлектронных систем» / Под общ. ред. академика РАН А.Л. Стемпковского. М.: ИППМ РАН. 2012. С. 83-88.

[6] Литвинов Е.И., Жихарев Г.Ю., Шагурин И.И. Возможности использования виртуальных платформ для верификации RTL-моделей сложно функциональных блоков в составе «Систем на кристалле» // Сб. трудов «Проблемы разработки перспективных микро- и наноэлектронных систем» / Под общ. ред. академика РАН А.Л. Стемпковского.Ч. 2. М.: ИППМ РАН. 2014. С. 51-56.

[7] Erickson J. TLM-Driven Design and Verification – Time For a Methodology Shift // Cadence Design Systems, Inc. URL: https://www.cadence.com/rl/Resources/white_papers/tlm-wp.pdf (дата обращения 31.03.2016).

[8] Черников В.М., Виксне П.Е., Шелухин А.М., Черников А.В., Косоруков Д.Е. Новый отечественный процессор обработки сигналов 1879ВМ4 семейства NeuroMatrix® // Сб. трудов «Проблемы разработки перспективных микро- и наноэлектронных систем» / Под общ. ред. академика РАН А.Л. Стемпковского. М.: ИППМ РАН. 2010. С. 241-246.

[9] Панфилов А.П., Косоруков Д.Е., Эйсымонт А.Л., Осипов В.Г., Черников В.М., Виксне П.Е., Шелухин А.М. Система на кристалле 1879ХК1 для цифровой обработки аналоговых сигналов в радиотехнических системах и спутниковой навигации // Сб. трудов «Проблемы разработки перспективных микро- и наноэлектронных систем» / Под общ. ред. академика РАН А.Л. Стемпковского. М.: ИППМ РАН. 2010. С. 221-226.

[10] Benini L., Bertozzi D., Bruni D. Legacy SystemC Co-Simulation of Multi-Processor Systems-on-Chip // IEEE International Conference on Computer Design: VLSI in Computers and Processor. 2002. P. 494–499.

[11] PPDL. URL: http://sk.ru/net/1120348/ (дата обращения 31.03.2016).

[12] Слепов А.Б. Разработка потактовой поведенческой модели системы на кристалле на языке C++ // Сб. трудов «Проблемы разработки перспективных микро- и наноэлектронных систем» / Под общ. ред. академика РАН А.Л. Стемпковского. М.: ИППМ РАН. 2010. С. 450-453.

[13] Bjorn F. Fast Cycle-Approximate Instruction Set Simulation // 11th International Workshop on Software: Compilers for Embedded Systems (SCOPES'08). 2008. P. 69-78.

[14] Беляев А.А. Принципы построения и разработка DSP-ядер с оптимальным по производительности конвейером для вычислительных и управляющих систем: автореф. дисс. ... канд. техн. наук: 05.13.05. – М. – 2010. – 25 с.

System level design of the DSP processor IP-core with NeuroMatrix architecture

V.V. Dementev

JSC STC «Module», v.dementev@module.ru

***Keywords* — instruction set simulator, cycle-accurate model, pipeline.**

ABSTRACT

Program (behavioral) models are a popular tool for exploration design space of system on chip, and software development. There is a large gap between the speed and accuracy of the functional instruction set simulators (ISS) and cycle-accurate models (CAS). Presence of processor cores causes necessity to maintain the parallel development of software and hardware and to conduct joint verification and debugging using one common behavioral model. The increasing complexity of VLSI and embedded software requires simulation technology, which has modeling speed higher than RTL can give and sufficient accuracy for assessment of system performance. The degree of detail of a model is determined by the level of abstraction, at which the model is applied, and its intended purpose. The report discusses usage of program models at the system design level (ESL) of processor IP-cores with NeuroMatrix architecture that are designed for DSP. Program models used at the system level should allow: debug software, evaluate the performance, explore design space and be reused. More accurate models of pipelined execution of commands are developed based on the ISS, which is improved to the required precision through the implementation of the algorithm of the pipeline working in C++/SystemC. To develop the ISS, sufficient information can be obtained from the programming reference manual. However, to develop the CAS, this information is not enough. The required information can be obtained during simulation at RTL level. Accuracy of the program model should be sufficient for real-time task.

REFERENCES

[1] Steshenko V., Rutkevich A., Gladkova E., Shishkin G., Voronov D. The design of VLSI type “system on chip”. The design route. The synthesis scheme. Part 1. Jelektronnye komponenty, 2009, no. 11, pp. 54–57 (in Russian).

[2] Evtushenko N., Nemudrov V., Syrcov I. The methodology of designing systems-on-chip. Basic principles, methods, software. Jelektronika: Nauka, Tehnologija, Biznes, 2003, no. 6, pp. 7–11 (in Russian).

[3] Nepomnjashhij O.V., Hnykin A.V. Design analysis of computer systems on a chip. Issledovatel'skij naukograd, 2012, no. 1, pp. 42-46 (in Russian).

[4] Nepomnjashhij O.V. , Legalov A.I., Sirotina N.Ju. Marshruty i tehnologii arhitekturno-nezavisimogo proektirovanija pri razrabotke slozhnyh odnokristal'nyh shem special'nogo naznachenija - *The routes and technology architecture-independent design in the development of complex on-chip circuits for special purposes.* Available at: http://ikit.sfu-kras.ru/files/ikit/8_Marshrut_proektirovania.pdf (accessed 31.03.2016).

[5] Putrja F.M. Features use of object-oriented programming the SystemVerilog for functional verification of multicore SoC. Trudy IPPM RAN «Problemy razrabotki perspektivnyh mikro- i nanojelektronnyh sistem» - *Proc. of IPPM RAS “Problems of development of perspective micro and nanoelectronic systems”*, 2012, pp. 83-88 (in Russian).

[6] Litvinov E.I., Zhiharev G.Ju., Shagurin I.I. Use of virtual platforms for verification of RTL models of difficult function blocks in the "Systems on chip". Trudy IPPM RAN «Problemy razrabotki perspektivnyh mikro- i nanojelektronnyh sistem» - *Proc. of IPPM RAS “Problems of development of perspective micro and nanoelectronic systems”*, 2014, no 2, pp. 51-56 (in Russian).

[7] Erickson J. “*TLM-Driven Design and Verification – Time For a Methodology Shift”*. Available at: https://www.cadence.com/rl/Resources/white_papers/tlm-wp.pdf (accessed 31.03.2016).

[8] Chernikov V.M., Viksne P.E., Sheluhin A.M., Chernikov A.V., Kosorukov D.E. A new domestic processor of signal processing 1879BM4 NeuroMatrix family. Trudy IPPM RAN «Problemy razrabotki perspektivnyh mikro- i nanojelektronnyh sistem» - *Proc. of IPPM RAS “Problems of development of perspective micro and nanoelectronic systems”*, 2010, pp. 241-246 (in Russian).

[9] Panfilov A.P., Kosorukov D.E., Jejsymont A.L., Osipov V.G., Chernikov V.M., Viksne P.E., Sheluhin A.M. System-on-chip 1879XK1 for digital processing of analog signals in radio systems and satellite navigation. Trudy IPPM RAN «Problemy razrabotki perspektivnyh mikro- i nanojelektronnyh sistem» - *Proc. of IPPM RAS “Problems of development of perspective micro and nanoelectronic systems”*, 2010., pp. 221-226 (in Russian).

[10] Benini L., Bertozzi D., Bruni D. *“Legacy SystemC Co-Simulation of Multi-Processor Systems-on-Chip”. Proc. of ICCD'02*, 2002, pp. 494–499.

[11] PPDL. Available at: http://sk.ru/net/1120348/ (accessed 31.03.2016) (in Russian).

[12] Slepov A.B. Development of cycle-accurate behavioral model of the system on chip in C++. Trudy IPPM RAN «Problemy razrabotki perspektivnyh mikro- i nanojelektronnyh sistem» - *Proc. of IPPM RAS “Problems of development of perspective micro and nanoelectronic systems”*, 2010, pp. 450-453 (in Russian).

[13] Bjorn F. *“Fast Cycle-Approximate Instruction Set Simulation”. Proc. of 11th Int. Workshop on Software “Compilers for Embedded Systems”*, 2008, pp. 69-78 (SCOPES'08).

[14] Beljaev A.A. Principy postroenija i razrabotka DSP-jader s optimal'nym po proizvoditel'nosti konvejerom dlja vychislitel'nyh i upravljajushhih sistem. Dokt, Diss. - The principles of construction and development of DSP cores with the optimal performance of the pipeline for computing and control systems. Doct. thesis. Moscow, 2010. 25 p. (in Russian).