УДК 004.942

Моделирование и верификация коммуникационных фабрик при проектировании систем на кристалле

А.Н. Готманов[1], М.А. Кишиневский[2], С. Чэттерджи[3]

[1]Strategic CAD Labs, Intel Corporation, Москва, alexander.gotmanov@intel.com

[2]Strategic CAD Labs, Intel Corporation, Хиллсборо, США

[3]Two Sigma Investments, LLC, Нью-Йорк, США

***Аннотация* — Один из способов убедиться в правильной работе системы на ранних этапах проектирования – построить абстрактную модель и провести формальное доказательство ее корректности. В статье рассмотрена задача верификации моделей коммуникационных фабрик, получающихся композицией небольшого числа базовых микроархитектурных блоков с простой семантикой. Мы покажем, что структурный анализ модели существенно упрощает доказательство ее свойств, в том числе свойства отсутствия зависаний. Предлагаемые методы успешно использовались для верификации микроархитектур промышленных систем на кристалле, тогда как другие подходы не дали результата или оказались слишком трудоемкими.**

***Ключевые слова* — моделирование, формальная верификация, xMAS, зависания, коммуникационные фабрики, микроархитектура, системы на кристалле.**

I. Введение

Коммуникационная фабрика (communication fabric) – это часть системы на кристалле, отвечающая за пересылку сообщений и пакетов данных между устройствами (процессорными ядрами, памятью, контроллерами внешних устройств и т.д.) [1]. Коммуникационные фабрики обладают сложной распределенной архитектурой с высоким уровнем параллелизма и конвейеризации для одновременной обработки множества транзакций, находящихся на разных стадиях исполнения. Ошибки, допущенные при проектировании коммуникационных фабрик, часто проявляются только при согласованном взаимодействии нескольких устройств, что существенно затрудняет их поиск. Методы имитационного моделирования позволяют проверить систему на небольшом подмножестве ее наиболее вероятных исполнений, но не гарантируют правильной работы в остальных случаях.

Мы предлагаем альтернативный подход, основанный на формальной верификации. Для этого сначала создается абстрактная микроархитектурная модель системы, а затем проводится автоматизированное доказательство ее корректности. Модели строятся на языке xMAS [2] путем композиции простых базовых блоков. Корректность определяется набором свойств, заданных предложениями линейной темпоральной логики [3]. Стандартными средствами (интерполяция [4], k-шаговая индукция [5], PDR [6]) удается обосновать корректность только очень небольших моделей xMAS. Для верификации промышленных микроархитектур мы разработали методы анализа, позволяющие ускорить доказательство, используя высокоуровневую структуру моделей, или свести сложную задачу (например, поиск зависаний) к существенно более простой.

В разделе II мы более подробно расскажем о языке xMAS, его семантике и свойствах, а также рассмотрим примеры моделей. В разделе III кратко рассмотрена задача автоматической генерации и доказательства некоторых типов свойств. Проблема зависаний и метод ее решения на основе стационарных переменных разобраны в разделе IV. Раздел V содержит результаты экспериментов. Заключение и план дальнейшей работы приведены в разделе VI.

II. Моделирование микроархитектуры

Определим набор стандартных блоков, далее называемых *примитивами* (рис. 1), поддерживающих простой протокол передачи данных. Выход одного примитива (называемого *инициатором*) соединяется с входом другого примитива (называемого *получателем*) однонаправленным *каналом*, передающим значения некоторого конечного типа. Канал u типа α (обозначение: $u:\alpha$) содержит два управляющих сигнала $\mathrm{u.irdy}$ и $\mathrm{u.trdy}$, указывающих готовность инициатора (“initiator ready”) и получателя (“target ready”), соответственно. Передача пакета происходит при $\mathrm{u.xfer} = (\mathrm{u.irdy} \cdot \mathrm{u.trdy}) = \mathbf{True}$. Сигнал данных $\mathrm{u.data}$ определяет содержимое передаваемого пакета и имеет тип α. Сеть из примитивов, соединенных каналами, назовем *моделью xMAS* (от eXecutable MicroArchitectural Specification).

Построение моделей путем простой композиции стандартных блоков обладает рядом заметных преимуществ. Бесшовное соединение примитивов каналами позволяет полностью избежать таких проблем, как потеря пакетов из-за ошибок в связывающей логике. Модели обладают интуитивным графическим представлением, подходящим для объяснения и документирования особенностей микроархитектуры. Наличие

высокоуровневой структуры существенно облегчает формальный анализ свойств модели.

A. *Примитивы*

Базовый набор примитивов xMAS показан на рис. 1. *Исток* (*source*) порождает пакеты с заданным значением e. В зависимости от типа истока, очередной пакет может появляться на каждом такте (*eager*), с недетерминированной задержкой или не появляться никогда (*dead*). Недетерминированный исток необходим для построения абстрактных моделей и имеет две разновидности: справедливый исток (*fair*) допускает только конечные задержки между пакетами; несправедливый исток (*unfair*) допускает бесконечную задержку, т.е. очередной пакет может никогда не появиться. В процессе обработки пакеты хранятся в *очередях* (*queue*) конечного размера, реализующих дисциплину FIFO. Новый пакет может быть помещен в очередь только при наличии в ней свободного места, а задержка прохождения пакета через пустую очередь равна одному такту. Примитив *преобразования* (*func*) изменяет значение пакета, применяя к нему функцию f. Пакет на выходе *барьера* (*join*) появляется только при наличии пакета на каждом из его входов, при этом выходной пакет вычисляется функцией h от входных пакетов. Входной пакет примитива *форк* (*fork*) преобразуется в два выходных пакета с помощью функций f и g. Примитив *ветвления* (*switch*) вычисляет два предиката s_a и s_b от входного пакета, и в зависимости от их значений перенаправляет пакет на один из двух выходов (одновременное обращение предикатов в **True** или **False** не допускается). При *слиянии* (*merge*) входные пакеты передаются на выход без изменений. Если два пакета поступают на входы слияния одновременно, посылается один из них, а второй задерживается. Для выбора между пакетами используется справедливый алгоритм арбитража[1]. Допускается слияние с произвольным числом входных каналов. *Сток* (*sink*) поглощает пакеты, завершая их обработку в модели. Аналогично истоку, существует четыре разновидности стока (*eager*, *fair*, *unfair*, *dead*).

Семантика примитива xMAS задается системой уравнений, связывающих значения сигналов irdy, trdy и data на соединенных с примитивом каналах. Например, форк с входом i и выходами a, b, параметризованный функциями f, g, описывается соотношениями

$$\begin{aligned} &\mathrm{a.data} := f(\mathrm{i.data}),\\ &\mathrm{b.data} := g(\mathrm{i.data}),\end{aligned}$$

$$\begin{aligned} &\mathrm{a.irdy} := \mathrm{i.irdy} \cdot \mathrm{b.trdy},\\ &\mathrm{b.irdy} := \mathrm{i.irdy} \cdot \mathrm{a.trdy},\\ &\mathrm{i.trdy} := \mathrm{a.trdy} \cdot \mathrm{b.trdy}.\end{aligned}$$

Такое определение гарантирует, что передачи на каналах i, a, b происходят одновременно. Использование оператора ":=" вместо отношения равенства подчеркивает функциональный характер зависимостей.

В качестве еще одного примера рассмотрим семантику примитива слияния по круговому алгоритму (round-robin) [1] с входами a, b и выходом o[2]:

$$\mathrm{o.data} := \begin{cases}\mathrm{a.data}, & \mathrm{s}=0,\\ \mathrm{b.data}, & \mathrm{s}=1,\end{cases}$$

$$\begin{aligned} &\mathrm{o.irdy} := \mathrm{a.irdy} \cdot \mathrm{b.irdy},\\ &\mathrm{a.trdy} := \mathrm{a.irdy} \cdot \mathrm{o.trdy} \cdot (\mathrm{s}=0),\\ &\mathrm{b.trdy} := \mathrm{b.irdy} \cdot \mathrm{o.trdy} \cdot (\mathrm{s}=1),\end{aligned}$$

$$\mathrm{s} := \begin{cases}0, & \mathrm{a.irdy} \cdot \neg \mathrm{b.irdy},\\ 1, & \neg \mathrm{a.irdy} \cdot \mathrm{b.irdy},\\ (\mathbf{pre}(\mathrm{s})+1)\ \mathbf{mod}\ 2, & \mathbf{pre}(\mathrm{o.irdy} \cdot \mathrm{o.trdy}),\\ \mathbf{pre}(\mathrm{s}), & \mathbf{True}.\end{cases}$$

В уравнениях используется дополнительный внутренний сигнал s со значениями {0,1} и три регистровые переменные, по одной на каждое применение оператора **pre**. Условия в выражении для сигнала s проверяются поочередно. Слияние с n входами определяется аналогично.

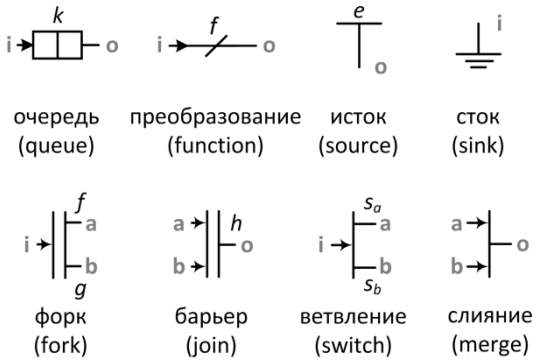

Рис. 1. Примитивы xMAS

Уравнения для очереди наиболее сложны, и мы их опускаем за недостатком места. Заметим только, что очередь q обладает внутренними сигналами q.num (число пакетов в очереди) и q.front (значение пакета в голове очереди). Полный список уравнений примитивов приведен в [2].

Используя уравнения примитивов, любой модели xMAS можно поставить в соответствие синхронную логическую сеть, называемую *синхронной моделью*. *Выполнением* (или *исполнением*) модели xMAS назовем любую допустимую бесконечную последовательность S состояний ее синхронной модели. Значения регистровых переменных в начальном состоянии считаются заданными (в частности, все очереди пусты). Исполнение, соблюдающее условия справедливости

[1] Алгоритм арбитража называется справедливым, если на каждом (бесконечном) исполнении модели входящий запрос может проиграть только конечное число раундов.

[2] Используемый в уравнениях оператор **pre** возвращает значение сигнала на предыдущем цикле. В начальный момент времени **pre** равен значению по умолчанию (0 для целых чисел, **False** для логических выражений).

для истоков и стоков, называется *справедливым*. Пара S, j, состоящая из исполнения S и натурального индекса j, обозначает суффикс исполнения S, начинающийся с j-го состояния. Начальное состояние имеет индекс 0.

Мы будем пользоваться линейной темпоральной логикой (Linear Temporal Logic, LTL) [3]. Темпоральный оператор "**F**" читается как "однажды", "**G**" – "всегда", "**X**" – "в следующий момент времени". $S \vDash \phi$ означает, что формула ϕ истинна в каждом состоянии исполнения S. Для модели M, $M \vDash \phi$ означает, что формула ϕ истинна в каждом исполнении M.

B. Устойчивость

Диаграмма состояний протокола передачи данных xMAS показана на рис. 2. Запрещенные переходы (например, из *FwRetry* в *Inactive*) гарантируют, что однажды начатая передача пакета не может быть отменена. Протокол, обладающий таким свойством, назовем *устойчивым*. Семантика примитивов xMAS определена так, чтобы гарантировать устойчивость на всех каналах в модели [7].

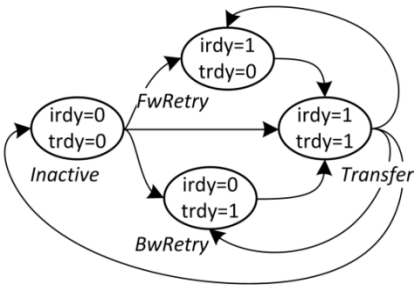

Рис. 2. Диаграмма состояний протокола xMAS. Также допустимы переходы из каждого состояния в себя

Понятие устойчивости можно определить в более общей форме для двух произвольных событий A и B:

$$A \rhd B \equiv \mathbf{G}(A \cdot \neg B \rightarrow \mathbf{X}A).$$

Интуитивно, устойчивость означает, что событие A, наступив однажды, будет повторяться на каждом такте до наступления события B. Полезно также ввести обозначение для взаимной устойчивости двух событий: $A \diamond B \equiv (A \rhd B) \cdot (B \rhd A)$. Устойчивость протокола передачи данных на канале $u{:}\alpha$ может быть выражена соотношениями:

$$\mathrm{u.irdy} \diamond \mathrm{u.trdy}, \quad \forall x \in \alpha.(u = x \rhd \mathrm{u.trdy}),$$

где "$u = x$" используется как сокращение для "$\mathrm{u.irdy} \cdot (\mathrm{u.data} = x)$".

C. Примеры моделей

На рис. 3 показан пример простой модели M_1, состоящей из истока, стока и двух очередей. Пакеты с 8-битным значением 0 порождаются истоком, проходят последовательно очереди q_1 и q_2 (в каждой очереди пакет находится не менее одного такта) и поглощаются стоком. Заметим, что исток и сток являются справедливыми, т.е. порождают и поглощают пакеты через произвольные конечные интервалы времени. В этих условиях зависания (прекращения активности) в модели M_1 быть не может.

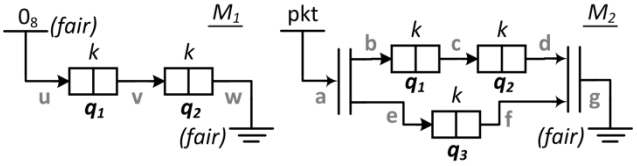

Рис. 3. Примеры моделей

В модели M_2 на рис. 3 каждый новый пакет удваивается, проходя через форк. Одна копия пакета попадает в цепочку из двух очередей q_1, q_2, вторая – в очередь q_3. Барьер на выходе снова собирает две копии пакета вместе и передает результат стоку. Очевидно, что общее число пакетов в верхней цепи q_1, q_2 равно числу пакетов в очереди q_3:

$$\mathrm{q_1.num} + \mathrm{q_2.num} = \mathrm{q_3.num}. \qquad (1)$$

Модель M_2 также свободна от зависаний.

III. Инварианты

Простейшие свойства моделей xMAS могут представлять серьезную трудность для стандартных алгоритмов верификации. Например, в модели M_1 на рис. 3 выполняется соотношение

$$\mathbf{G}(\mathrm{w.irdy} \rightarrow (\mathrm{w.data} = 0_8)), \qquad (2)$$

т.е. все пакеты, передаваемые на канале w, имеют 8-битное значение 0. Доказательство свойства (2) должно использовать тот факт, что все пакеты в очередях q_1 и q_2 также имеют значение 0_8. Мы условно обозначим это через

$$\mathbf{G}(\forall x \in q_i.x = 0_8), i = 1{,}2. \qquad (3)$$

Многие алгоритмы верификации испытывают трудности при доказательстве предложения (2), если соотношения (3) не заданы явно. В [2] показано как, зная исходный инвариант (2), построить дополнительные усиливающие инварианты (3) для произвольной модели xMAS и улучшить производительность средств формальной верификации.

Примером другого важного типа свойств является *линейный инвариант* (1). Это соотношение баланса числа пакетов аналогичное уравнениям Кирхгофа в электрических сетях. Знание уравнения (1) крайне полезно для доказательства других нетривиальных свойств системы, в частности свойства отсутствия зависаний. Линейные инварианты для больших моделей xMAS могут содержать десятки очередей, а их поиск вручную оказывается весьма затруднительным. Глобальные свойства сложной электрической схемы можно вывести, пользуясь простым законом локального сохранения заряда (сумма токов, втекающих в узел, равна сумме токов, вытекающих из узла). Аналогично, уравнение (1) может быть получено на основе "законов сохранения пакетов", которым удовлетворяют примитивы xMAS. Например, для примитива форк на рис. 1 выполнены равенства

$$n_i = n_a = n_b,$$

где n_i, n_a, n_b равно общему числу передач на каналах i, a, b с начального момента времени. Закон для очереди имеет вид

$$n_i = \mathrm{q.num} + n_o,$$

т.е. число пакетов, вошедших в очередь, равно числу пакетов вышедших из очереди, плюс число пакетов, находящихся в очереди. Рассматривая аналогичные линейные уравнения для всех примитивов в модели (их полный список можно найти в [2]) и упрощая систему методом Гаусса, можно находить соотношения баланса пакетов в очередях автоматически.

IV. Поиск зависаний

Интуитивное понятие "зависания" связано с нежелательным прекращением активности. Рассмотрим модель M_1 на рис. 3. Допустим, что в некотором исполнении S канал v перестает передавать пакеты, т.е. выполнено $\mathbf{Inactive}(v) = \mathbf{FG}(\neg \mathrm{v.xfer})$. Это может быть вызвано тем, что очередь q_1 опустела или очередь q_2 переполнилась. Более формально, для каждого исполнения S модели M_1 выполено

$$\mathbf{Inactive}(v) \to \mathbf{Empty}(q_1) + \mathbf{Full}(q_2),$$
$$\mathbf{Empty}(q_1) = \mathbf{FG}(\mathrm{q_1.num} = 0),$$
$$\mathbf{Full}(q_2) = \mathbf{FG}(\mathrm{q_2.num} = k).$$

Очередь q_1 может опустеть, только если исток перестал посылать пакеты на канале u. Очередь q_2 может переполниться, только если сток перестал принимать пакеты на канале w:

$$\mathbf{Empty}(q_1) \to \mathbf{Idle}(u), \quad \mathbf{Full}(q_2) \to \mathbf{Block}(w),$$
$$\mathbf{Idle}(u) = \mathbf{FG}(\neg \mathrm{u.irdy}), \quad \mathbf{Block}(w) = \mathbf{FG}(\neg \mathrm{w.trdy}).$$

Из приведенных соотношений следует, что $\mathbf{Inactive}(v) \to \mathbf{Idle}(u) + \mathbf{Block}(w)$, т.е. зависание на канале v может быть вызвано только отсутствием пакетов на канале u или блокировкой канала w. Это, очевидно, невозможно в силу справедливости истока и стока. Предположение об их справедливости можно выразить как $\mathbf{Fair} = \neg\mathbf{Idle}(u) \cdot \neg\mathbf{Block}(w)$. Окончательный вывод можно сформулировать как

$$M_1 \vDash \mathbf{Inactive}(v) \to \neg\mathbf{Fair}.$$

Таким образом, зависание возможно только на несправедливом исполнении.

Условия $\mathbf{FG}(A)$, определяющие поведение модели на бесконечности, будем называть *стационарными переменными*. Значение стационарной переменной зависит только от выбора исполнения S. Рассуждая о стационарных переменных для каналов и примитивов модели M_1, мы доказали отсутствие зависания на канале v. Доказательство опирается на известные свойства примитивов и законы булевой алгебры.

A. *Определение зависания*

В рассмотренном примере зависание определяется как отсутствие передач на канале v. На практике ситуация, когда агент перестает посылать пакеты в коммуникационную фабрику, не должна рассматриваться как зависание. Определим *отсутствие зависания* на канале u как

$$\mathbf{Live}(u) = \mathbf{G}(\mathrm{u.irdy} \to \mathbf{F}\mathrm{u.trdy}),$$

т.е. за сигналом $\mathrm{u.irdy}$ через конечное число тактов всегда следует сигнал $\mathrm{u.trdy}$. При отсутствии пакетов сигнал $\mathrm{u.irdy}$ равен **False**, и канал u автоматически удовлетворяет свойству **Live**. *Наличие зависания* определяется как отрицание $\mathbf{Live}(u)$:

$$\mathbf{Dead}(u) = \neg\mathbf{Live}(u) = \mathbf{F}(\mathrm{u.irdy} \cdot \mathbf{G}\neg\mathrm{u.trdy}).$$

Заметим, что зависание на канале может возникать в одном исполнении S_1 и отсутствовать в другом исполнении S_2. Будем говорить, что на канале u *возможно зависание*, если существует справедливое исполнение S модели M такое, что $S \vDash \mathbf{Dead}(u)$. Для устойчивого протокола определения свойств **Live** и **Dead** можно упростить.

Теорема 1. Пусть $\mathrm{u.irdy} \not\Rightarrow \mathrm{u.trdy}$. Тогда

$$\mathbf{Live}(u) = \mathbf{FG}(\neg\mathrm{u.irdy}) + \mathbf{GF}(\mathrm{u.trdy}),$$
$$\mathbf{Dead}(u) = \mathbf{GF}(\mathrm{u.irdy}) \cdot \mathbf{FG}(\neg\mathrm{u.trdy}).$$

Для каждого справедливого истока с выходом u и каждого справедливого стока с входом v определим условия

$$\mathbf{FSnd}(u) = \mathbf{GF}(\mathrm{u.irdy}), \quad \mathbf{FRcv}(u) = \mathbf{GF}(\mathrm{u.trdy}).$$

Через $\mathbf{Fair}_M$ обозначим конъюнкцию таких условий для всех справедливых истоков и стоков в модели M. Справедливость исполнения S эквивалентна условию $S \vDash \mathbf{Fair}_M$. Модель M свободна от зависаний на канале u тогда и только тогда, когда

$$M \vDash \mathbf{Fair}_M \to \neg\mathbf{Dead}(u).$$

Таблица 1

Стационарные уравнения для примитивов xMAS

Преобразование с входом $i\colon \alpha$, выходом $o\colon \beta$ и функцией $f\colon \alpha \to \beta$. Для каждого $y \in \beta$

$$\begin{aligned}\mathbf{Block}(i) &= \mathbf{Block}(o),\\ \mathbf{Idle}^{y}(o) &= \prod_{x:f(x)=y} \mathbf{Idle}^{x}(i).\end{aligned}$$

Очередь размера k с входом $i\colon \alpha$ и выходом $o\colon \alpha$. Для каждого $x \in \alpha$

$$\begin{aligned}\mathbf{Block}(i) &= \mathbf{Full}(q),\\ \mathbf{Idle}^{x}(o) &= \mathbf{Idle}^{x}(q),\\ \mathbf{Empty}(q) &\to \neg\mathbf{Full}(q),\\ \mathbf{Full}(q) &\to \mathbf{Block}(o),\\ \mathbf{Empty}(q) &= \mathbf{Idle}(q),\\ \mathbf{Block}(o) &\to \mathbf{Idle}(i) + \mathbf{Full}(q),\\ \neg\mathbf{Block}(o) &\to \mathbf{Idle}^{x}(i) = \mathbf{Idle}^{x}(q).\end{aligned}$$

Форк с входом $i\colon \alpha$, выходами $a\colon \alpha$, $b\colon \alpha$ и тождественными функциями f, g. Для каждого $x \in \alpha$

$$\begin{aligned}\mathbf{Block}(i) &= \mathbf{Block}(a) + \mathbf{Block}(b),\\ \mathbf{Idle}^{x}(a) &= \mathbf{Idle}^{x}(i) + \mathbf{Block}(b),\\ \mathbf{Idle}^{x}(b) &= \mathbf{Idle}^{x}(i) + \mathbf{Block}(a).\end{aligned}$$

Ограниченный барьер с входами $a\colon \alpha$, $b\colon \beta$, выходом $o\colon \alpha$ и тождественной функцией h. Для каждого $x \in \alpha$

$$\begin{aligned}\mathbf{Idle}^{x}(o) &= \mathbf{Idle}^{x}(a) + \mathbf{Idle}(b),\\ \mathbf{Block}(a) &= \mathbf{Block}(o) + \mathbf{Idle}(b),\\ \mathbf{Block}(b) &= \mathbf{Block}(o) + \mathbf{Idle}(a).\end{aligned}$$

Ветвление с входом $i\colon \alpha$, выходами $a\colon \alpha$, $b\colon \alpha$ и предикатами $s_a(x)$, $s_b(x)$. Для каждого $x \in \alpha$

$$\begin{aligned}\mathbf{Block}(i) &= \mathbf{Idle}(i) +\\ &\quad + \mathbf{Block}(a)\cdot\mathbf{Idle}^{s_b(x)}(i) +\\ &\quad + \mathbf{Block}(b)\cdot\mathbf{Idle}^{s_a(x)}(i),\\ \mathbf{Idle}^{x}(a) &= \neg s_a(x) + \mathbf{Idle}^{x}(i),\\ \mathbf{Idle}^{x}(b) &= \neg s_b(x) + \mathbf{Idle}^{x}(i).\end{aligned}$$

Слияние с входами $a\colon \alpha$, $b\colon \alpha$ и выходом $o\colon \alpha$. Для каждого $x \in \alpha$

$$\begin{aligned}\mathbf{Block}(a) &= \mathbf{Idle}(a) + \mathbf{Block}(o),\\ \mathbf{Block}(b) &= \mathbf{Idle}(b) + \mathbf{Block}(o),\\ \mathbf{Idle}^{x}(o) &= \mathbf{Idle}^{x}(a)\cdot\mathbf{Idle}^{x}(b) + \mathbf{Idle}^{x}(a)\cdot\mathbf{Sel}_a(m) + \mathbf{Idle}^{x}(b)\cdot\mathbf{Sel}_b(m),\\ \mathbf{Sel}_a(m) &= \mathbf{Idle}(o) + \neg\mathbf{Idle}(a)\cdot(\mathbf{Idle}(b) + \mathbf{Block}(o)\cdot\neg\mathbf{Sel}_b(m)),\\ \mathbf{Sel}_b(m) &= \mathbf{Idle}(o) + \neg\mathbf{Idle}(b)\cdot\big(\mathbf{Idle}(a) + \mathbf{Block}(o)\cdot\neg\mathbf{Sel}_a(m)\big).\end{aligned}$$

B. Стационарные переменные

Поведение модели xMAS на бесконечности можно описать с помощью небольшого набора стационарных переменных. Для каждого канала $u\colon \alpha$ определим

$$\mathbf{Idle}(u) = \mathbf{FG}(\neg \mathrm{u.irdy}), \quad \mathbf{Block}(u) = \mathbf{FG}(\neg \mathrm{u.trdy}).$$

Чтобы учесть значение данных на канале, введем также уточнение переменной **Idle**:

$$\mathbf{Idle}^{x}(u) = \mathbf{FG}(\mathrm{u.irdy} \to (\mathrm{u.data} \neq x)), \quad x \in \alpha.$$

Переменная $\mathbf{Idle}^{x}(u)$ описывает отсутствие пакета со значением x на канале u. Мы также будем пользоваться сокращением $\mathbf{Idle}^{p(x)}(u) = \prod_{x:p(x)} \mathbf{Idle}^{x}(u)$. Заметим, что $\mathbf{Idle}(u) = \mathbf{Idle}^{\mathbf{True}}(u)$.

Используя теорему 1, можно выразить условие зависания $\mathbf{Dead}(u)$ через стационарные переменные:

$$\mathbf{Dead}(u) = \neg\mathbf{Idle}(u)\cdot\mathbf{Block}(u).$$

Аналогично выражаются условия справедливости:

$$\mathbf{FSnd}(u) = \neg\mathbf{Idle}(u), \quad \mathbf{FRcv}(v) = \neg\mathbf{Block}(v).$$

Для примитива слияния m с выходным каналом o и входным каналом a положим

$$\mathbf{Sel}_a(m) = \mathbf{FG}(\mathrm{o.irdy} \to (\mathrm{m.s} = i_a)),$$

где i_a – целочисленный номер входа a в слиянии m. Наконец, для очереди q размера k с выходным каналом $o\colon \alpha$,

$$\begin{aligned}\mathbf{Full}(q) &= \mathbf{FG}(\mathrm{q.num} = k),\\ \mathbf{Empty}(q) &= \mathbf{FG}(\mathrm{q.num} = 0),\\ \mathbf{Idle}^{x}(q) &= \mathbf{FG}\big((\mathrm{q.num} = 0) + (\mathrm{q.front} \neq x)\big).\end{aligned}$$

C. Стационарные уравнения

Стационарные переменные в модели xMAS тесно связаны друг с другом. Для каждого примитива мы укажем уравнения, ограничивающие возможные значения стационарных переменных в его окрестности (включая стационарные переменные самого примитива). Систему, включающую уравнения для всех примитивов в модели, назовем *стационарной структурой модели* M и обозначим через $\mathbf{Struct}_M$.

Полный список уравнений приведен в табл. 1. Для примитива форк рассмотрен случай тождественных функций f и g. Использование нетождественных функций эквивалентно добавлению примитивов преобразования. Мы предполагаем, что все барьеры являются *ограниченными*, т.е. их функции h зависят только от одного из своих аргументов. Это сужает класс моделей xMAS, к которым применим метод анализа на основе стационарных уравнений. Однако многие встречающиеся на практике микроархитектуры могут быть описаны с использованием только ограниченной формы барьера.

Теорема 2. Для модели M с ограниченными барьерами верно, что $M \vDash \mathbf{Struct}_M$.

Теорема 2 обобщает рассуждение приведенное в начале раздела (на примере M_1, рис.3) для произвольной модели M. Пусть требуется проверить отсутствие зависания на канале u. Построим систему $\mathbf{Struct}_M$, выписывая уравнения из табл. 1 для каждого примитива в M. Затем добавим ограничения $\mathbf{Fair}_M$ и $\mathbf{Dead}(u)$, выраженные через стационарные переменные. Если система

$$\mathbf{Struct}_M \cdot \mathbf{Fair}_M \cdot \mathbf{Dead}(u) \qquad (4)$$

несовместна как система булевых уравнений, то $\mathbf{Struct}_M \cdot \mathbf{Fair}_M \to \neg\mathbf{Dead}(u)$, и условие $\mathbf{Dead}(u)$ не может быть выполнено на справедливом исполнении модели. Если же система совместна, то, анализируя значения стационарных переменных, можно получить представление о причине зависания.

D. Использование инвариантов

Метод стационарных уравнений может приводить к ложным контрпримерам, т.е. решение системы (4) может присваивать стационарным переменным значения, которые не реализуются ни в одном допустимом исполнении модели M. Этого можно избежать, приближая множество достижимых состояний модели инвариантами. Рассмотрим модель M_2 на рис. 3. Для нее система (4) имеет решение, в котором

$$\mathbf{Full}(q_1) = \mathbf{Full}(q_2) = \mathbf{Empty}(q_3) = \mathbf{True}. \quad (5)$$

Допустим, что существует некоторое исполнение S, удовлетворяющее равенствам (5). Тогда, начиная с некоторого момента времени j,

$$S, j \vDash (\mathrm{q}_1.\mathrm{num} = k) \cdot (\mathrm{q}_2.\mathrm{num} = k) \cdot (\mathrm{q}_3.\mathrm{num} = 0).$$

Последнее противоречит инварианту (1) модели M_2. Следовательно, такого исполнения S не существует, контрпример является ложным и должен быть исключен из рассмотрения.

Исключение ложных контрпримеров можно провести автоматически, если задан некоторый инвариант $\mathbf{Inv}_M$ модели M. Составим вспомогательную систему $\mathbf{Inf}_M$, содержащую для каждой используемой стационарной переменной $\mathbf{FG}A$ уравнение $\mathbf{FG}A \to A$. $\mathbf{Inf}_M$ формализует "предельный переход", показанный выше на примере.

Теорема 3. Для каждого исполнения S модели M существует суффикс S, j_∞ такой, что $S, j_\infty \vDash \mathbf{Inf}_M$.

Из теорем 2 и 3 следует, что для каждого справедливого исполнения S модели M

$$S, j_\infty \vDash \mathbf{Struct}_M \cdot \mathbf{Fair}_M \cdot \mathbf{Inf}_M \cdot \mathbf{Inv}_M.$$

Из несовместности усиленной (по сравнению с (4)) системы $\mathbf{Struct}_M \cdot \mathbf{Fair}_M \cdot \mathbf{Inf}_M \cdot \mathbf{Inv}_M \cdot \mathbf{Dead}(u)$ следует, что зависание на канале u невозможно. Правильный подбор инварианта $\mathbf{Inv}_M$ позволяет исключить все ложные контрпримеры. В большинстве практических случаев $\mathbf{Inv}_M$ можно положить равным конъюнкции автоматически сгенерированных линейных инвариантов (раздел II).

V. Результаты экспериментов

Метод стационарных уравнений применялся для поиска зависаний в различных моделях xMAS, включая модели промышленных коммуникационных фабрик. Были рассмотрены примеры с конвейеризацией запросов, многошаговой обработкой транзакций, упорядочиванием сообщений, виртуальными каналами, механизмами целостности памяти и т.д. Мы сравнили наш метод с верификацией моделей в среде ABC [8] (использовались интерполяция [4] и представление свойства отсутствия зависаний в виде инварианта [9]). Во всех случаях для улучшения сходимости к модели добавлялись автоматически сгенерированные инварианты [2]. Те же инварианты использовались для исключения ложных контрпримеров в методе стационарных уравнений.

Таблица 2

Результаты работы метода

Модель	p	q	T
SMF	75	24	4c
IOF	493	97	90c
CMF	710	88	690c

В табл. 2 приведены результаты работы нашего метода на трех моделях промышленных коммуникационных фабрик. Для каждого из примеров указано общее число примитивов в модели (p), число очередей (q) и время работы алгоритма до получения доказательства отсутствия зависаний (t). Размеры очередей не указаны, так как производительность метода от них зависит слабо. Во всех рассмотренных случаях работа алгоритма интерполяции в ABC не завершилась за 1 час.

VI. Заключение

В статье рассмотрен язык микроархитектурного моделирования xMAS и связанные с ним методы анализа. Модели xMAS имеют ясное графическое представление и помогают избежать многих типовых ошибок. Использование структурного анализа позволяет существенно ускорить формальную верификацию. Предложенный метод стационарных уравнений сводит поиск зависаний к обычной задаче выполнимости. С его помощью нам удалось провести анализ многих промышленных примеров, где другие подходы не дали результатов. Для моделей xMAS произвольного вида (с неограниченными барьерами) метод стационарных уравнений требует существенных модификаций.

Литература

[1] Dally W.J., Towles B. Principles and Practices of Interconnection Networks. Morgan Kaufmann Publishers, 2004. 550 p.

[2] Chatterjee S., Kishinevsky M. Automatic generation of inductive invariants from highlevel microarchitectural models of communication fabrics // Proc. CAV, LNCS. 2010. V. 6174. P. 321–338.

[3] Manna Z., Pnueli A. The Temporal Logic of Reactive and Concurrent Systems: Specification. Springer-Verlag, 1991. 462 p.

[4] McMillan K. Interpolation and SAT-based Model Checking // Proc. CAV, LNCS. 2003. V. 2725. P. 1–13.

[5] Sheeran M. et al. Checking Safety Properties Using Induction and a SAT-Solver // Proc. FMCAD, LNCS. 2000. V. 1954. P. 108–125.

[6] Bradley A. SAT-based Model Checking without Unrolling // Proc. VMCAI, LNCS. 2011. V. 6538. P. 70–87.

[7] Gotmanov A., Chatterjee S., Kishinevsky M. Verifying Deadlock-Freedom of Communication Fabrics // Proc VMCAI, LNCS. 2011. V. 6538. P. 214–231.

[8] ABC: A system for sequential synthesis and verification. http://www.eecs.berkeley.edu/~alanmi/abc/.

[9] Schuppan V., Biere A. Liveness Checking as Safety Checking // ENTCS. 2006. V. 149. № 1. P. 79–96.