УДК 004.942

Анализ задержек в микроархитектурных моделях коммуникационных фабрик

Ю.О. Викторов, А.Н. Готманов

Strategic CAD Labs, Intel Corporation, Москва, {yuriy.viktorov, alexander.gotmanov}@intel.com

***Аннотация* — В этой статье мы изложим теоретические основы метода анализа задержек в микроархитектурных моделях коммуникационных фабрик. Мы покажем, как получить верхнюю оценку задержки и верифицировать ее с помощью ранжирующих функций.**

***Ключевые слова* — качество обслуживания, формальная верификация, xMAS, системы на кристалле, коммуникационные фабрики, ранжирующие функции.**

I. Введение

Взаимодействие устройств в системе на кристалле реализуется *коммуникационной фабрикой* (*communication fabric*) [1]. Производительность системы определяется эффективностью разделения ограниченных аппаратных ресурсов конкурирующими процессами, а качество проектируемой архитектуры напрямую зависит от умения оценивать минимальный уровень обслуживания, который система гарантирует своим агентам.

В этой статье мы рассмотрим *задержку передачи* (*latency*) – интервал времени между отправлением данных источником и их получением в точке назначения. Имитационное моделирование позволяет получить хорошее представление о средних задержках. Однако поведение системы в наихудшем случае может сильно отличаться от среднего. Надежную оценку производительности можно получить с помощью формальной верификации.

Для применения средств верификации необходимо построить формальную модель системы. Язык xMAS [2] позволяет создавать наглядные модели коммуникационных фабрик путем композиции стандартных блоков. Ограничение задержки передачи пакета в модели xMAS (например, пакет попадает из точки A в точку B не более чем за N тактов) легко сформулировать в виде предложения линейной темпоральной логики (Linear Temporal Logic, LTL) [3]. Однако его доказательство часто оказывается за пределами возможностей современных алгоритмов верификации моделей (интерполяция [4], k-шаговая индукция [5], PDR [6]), так как требует рассмотрения порядка N последовательных тактов работы системы.

Перспективным представляется альтернативный подход, основанный на использовании ранжирующих функций [7]. С их помощью доказательство оценки на задержку можно свести к более простой задаче, для решения которой, как правило, достаточно 1-шаговой индукции.

Наш метод основан на формализации понятия задержки в рамках LTL. Мы покажем, как воспользоваться элементарными свойствами задержки для анализа времени выполнения многошаговых транзакций в моделях xMAS. Построение оценки на задержку представлено как вывод из гипотез в формальной теории и сопровождается параллельным построением ранжирующей функции. Таким образом, одновременно решаются задачи вычисления задержки и ее верификации. В этой статье мы изложим теоретические основы метода и представим предварительные результаты экспериментов на моделях небольшого размера.

В разделе II мы более подробно расскажем о расширениях языка xMAS, необходимых для моделирования качества обслуживания. Определив понятие задержки в разделе III, мы рассмотрим пример простой модели коммуникационной фабрики и покажем, как вычислить задержку обработки запроса. В разделе IV мы объясним, как провести формальную верификацию оценки с помощью ранжирующих функций. Раздел V содержит предварительные результаты экспериментов. Планы дальнейшей работы и заключение приведены в разделе VI.

II. Язык моделирования

Для описания микроархитектуры мы воспользуемся языком моделирования xMAS[1]. Модели xMAS состоят из *примитивов*, соединенных *каналами* для передачи пакетов с данными. Каждый канал u содержит два управляющих сигнала $u.irdy$ и $u.trdy$, указывающих готовность инициатора передачи (“initiator ready”) и ее получателя (“target ready”) соответственно. Пакет передается при $u.xfer = (u.irdy \cdot u.trdy) = \mathbf{True}$. В этой статье мы используем набор примитивов, показанный на рис. 1.

A. Моделирование задержек

Для анализа задержек необходимо расширить базовый набор примитивов xMAS. Например, стандартная реализация слияния использует круговой алгоритм

[1] Более подробное описание языка xMAS можно найти в [2] и в статье «Моделирование и верификация коммуникационных фабрик при проектировании систем на кристалле» данного сборника.

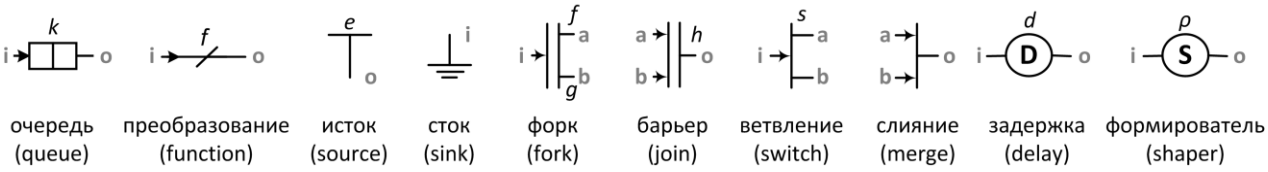

Рис. 1. Набор примитивов для моделирования задержек. Полужирным шрифтом набраны имена каналов, курсивом – параметры примитивов

арбитража (round-robin) [1]. Такое упрощение приемлемо для многих задач, например, для поиска тупиковых ситуаций или доказательства некоторых инвариантов. Однако при моделировании качества обслуживания целесообразно использовать несколько разновидностей примитива слияния, отличающихся алгоритмом арбитража.

Для моделирования задержек обработки введем новый примитив *ограниченной задержки* (*delay*). Поведение задержки величины d с входом i и выходом o задается уравнениями[2]:

$$\begin{aligned} \mathrm{i.trdy} &:= \mathrm{o.trdy} \cdot (\mathrm{cnt} = 0), \\ \mathrm{o.irdy} &:= \mathrm{i.irdy} \cdot (\mathrm{cnt} = 0), \end{aligned}$$

$$\mathbf{next}(\mathrm{cnt}) := \begin{cases} d, & \mathrm{i.irdy} \cdot \mathrm{i.trdy}, \\ \mathrm{cnt} - 1, & \mathrm{i.irdy} \cdot \neg \mathrm{i.trdy} \cdot (\mathrm{cnt} > 0), \\ \mathrm{cnt}, & \neg \mathrm{i.irdy} + \neg \mathrm{i.trdy} \cdot (\mathrm{cnt} = 0), \end{cases}$$

$$\mathbf{init}(\mathrm{cnt}) := d.$$

Целочисленная переменная cnt принимает значения в {0..d} и задает число тактов ожидания до передачи. Случайная задержка с верхней границей d моделируется аналогично.

Другой тип задержек создается *формирователем потока* (*shaper*). Формирователь гарантирует, что пакеты на выходе появляются не чаще, чем с заданной частотой ρ, задерживая при необходимости входные пакеты. Пусть $\rho = R/N$ для целых положительных чисел R и N. Тогда поведение примитива можно описать уравнениями:

$$\begin{aligned} \mathrm{i.trdy} &:= \mathrm{o.trdy} \cdot (\mathrm{bkt} \geq N - R), \\ \mathrm{o.irdy} &:= \mathrm{i.irdy} \cdot (\mathrm{bkt} \geq N - R), \end{aligned}$$

$$\mathbf{next}(\mathrm{bkt}) := \begin{cases} \mathrm{bkt} - N + R, & \mathrm{i.xfer}, \\ \mathbf{min}\{\mathrm{bkt} + R, N\}, & \neg \mathrm{i.xfer}, \end{cases}$$

$$\mathbf{init}(\mathrm{bkt}) := N.$$

Формирователь работает по принципу “leaky bucket”, увеличивая целочисленную переменную bkt на R на каждом такте. Входящий пакет задерживается, если текущее значение bkt меньше чем $N - R$. При передаче пакета переменная bkt уменьшается на N.

B. Пример модели

В завершение раздела рассмотрим подробнее пример модели на рис. 2. Пакет со значением P порождается истоком и ожидает срабатывания барьера на канале a. На второй вход барьера поступают управляющие пакеты (*токены*) от формирователя с частотой $1/2$. Это гарантирует, что на канале b за два такта работы появляется не более одного нового пакета. Далее пакет P проходит слияние, конкурируя с потоком пакетов Q и, выиграв арбитраж, попадает в очередь по каналу c. Заметим, что пакеты Q порождаются недетерминированным истоком, и частота их появления может быть произвольной. Покидая первую очередь по каналу d, пакеты P и Q проходят ветвление и перераспределяются по каналам e и u в различные очереди, на выходе которых пакеты задерживаются. Пакеты покидают модель по каналу h после повторного слияния потоков.

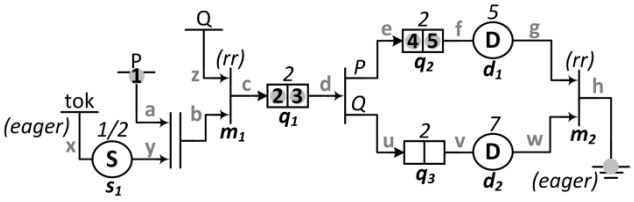

Рис. 2. Модель коммуникационной фабрики *F*. Все слияния работают по круговому алгоритму (*rr*). Цифрами 1-5 обозначены возможные положения пакета *P*

Модель на рис. 2 можно также интерпретировать как простейшую коммуникационную фабрику, принимающую запросы от двух агентов P и Q. Пакеты принимаются фабрикой по одному за такт, что моделируется слиянием на входе. Обработка запросов конвейеризирована. На первой ступени запросы обрабатываются последовательно и хранятся в общей очереди. На второй ступени запросы от разных агентов обрабатываются параллельно с различными задержками. Перед удалением из фабрики запросы проходят завершающую фазу по одному за такт, что соответствует слиянию на выходе. Общую задержку обработки запроса P коммуникационной фабрикой естественно определить как количество тактов с момента появления запроса на канале a до момента его передачи на канале h.

III. ЗАДЕРЖКА

A. Отношение отклика

Неформально задержку можно определить как количество тактов, протекающее между *предусловием* A и *постусловием* B, где A и B – произвольные события, связанные *отношением отклика*:

$$A \rightsquigarrow B \equiv \mathbf{G}(A \rightarrow \mathbf{F}B).$$

[2]Используемый в уравнениях оператор $\mathbf{next}(x)$ определяет значение регистровой переменной x на следующем такте; $\mathbf{init}(x)$ – в начальный момент времени.

Таблица 1

Правила вывода $\mathcal{L}$

$$\frac{}{\mathbf{Lat}[A \rightsquigarrow A] = 0}\ \text{(Ref-Pre)} \qquad \frac{}{\mathbf{Lat}[\mathbf{True} \rightsquigarrow_E E] = 0}\ \text{(Ref-Mod)} \qquad \frac{A \rhd B, \quad \mathbf{Lat}[A \rightsquigarrow_E B] \le l}{\mathbf{Lat}[A \rightsquigarrow_E A \cdot B] \le l}\ \text{(Persist)}$$

$$\frac{\mathbf{Lat}[A \rightsquigarrow_E B] \le k}{\mathbf{Lat}[A \cdot C \rightsquigarrow_E B] \le k}\ \text{(Pre-And)} \qquad \frac{\mathbf{Lat}[A \rightsquigarrow_E C] \le l, \quad \mathbf{Lat}[B \rightsquigarrow_E C] \le k}{\mathbf{Lat}[A + B \rightsquigarrow_E C] \le \mathbf{max}\{l,k\}}\ \text{(Pre-Or)} \qquad \frac{\mathbf{Lat}[A \rightsquigarrow_E B] \le l, \quad \mathbf{Lat}[A \rightsquigarrow_E B] \le k}{\mathbf{Lat}[A \rightsquigarrow_E B] \le \mathbf{min}\{l,k\}}\ \text{(Min)}$$

$$\frac{B \rhd C,\ C \rhd B, \quad \mathbf{Lat}[A \rightsquigarrow_E B] \le l, \quad \mathbf{Lat}[A \rightsquigarrow_E C] \le k}{\mathbf{Lat}[A \rightsquigarrow_E B \cdot C] \le \mathbf{max}\{l,k\}}\ \text{(Post-And)} \qquad \frac{\mathbf{Lat}[A \rightsquigarrow_E B] \le l}{\mathbf{Lat}[A \rightsquigarrow_E B + C] \le l}\ \text{(Post-Or)} \qquad \frac{A \rhd B, \quad B \rhd E, \quad \mathbf{Lat}[A \rightsquigarrow_E B] \le l, \quad \mathbf{Lat}[A \rightsquigarrow E] \le k}{\mathbf{Lat}[A \rightsquigarrow B \cdot E] \le l + k + k \cdot l}\ \text{(Iter)}$$

$$\frac{\mathbf{Lat}[A \rightsquigarrow_E B] \le l}{\mathbf{Lat}[A \rightsquigarrow_{E \cdot F} B] \le l}\ \text{(Mod-And)} \qquad \frac{\mathbf{Lat}[A \rightsquigarrow_E B] \le l, \quad \mathbf{Lat}[A \rightsquigarrow_F B] \le k}{\mathbf{Lat}[A \rightsquigarrow_{E+F} B] \le l + k}\ \text{(Mod-Or)} \qquad \frac{\mathbf{Lat}[A \rightsquigarrow_E B] \le l, \quad \mathbf{Lat}[B \rightsquigarrow_E C] \le k}{\mathbf{Lat}[A \rightsquigarrow_E C] \le l + k}\ \text{(Seq)}$$

$$\frac{}{\mathbf{Lat}[\mathbf{X}A \rightsquigarrow A] \le 1}\ \text{(Next)}$$

За событием A всегда следует событие B. Мы пользуемся стандартными обозначениями LTL [3]: $\mathbf{G}A$ – "всегда A", $\mathbf{F}A$ – "однажды A", $\mathbf{X}A$ – "в следующий момент времени A". Выражение ϕ^+ равно значению величины ϕ на следующем такте.

Определим *локальную задержку* $\mathbf{lat}[B]$, равную числу тактов между текущим моментом времени и следующим наступлением события B:

$$(\mathbf{lat}[B] \le k) \equiv \mathbf{F}^k B,$$

где $\mathbf{F}^k B = B + \mathbf{X}B + \cdots + \mathbf{X}^k B$, а $\mathbf{X}^k$ соответсвует k последовательным применениям темпорального оператора $\mathbf{X}$. Если событие B никогда не наступает, т.е. выполнено $\mathbf{G}\neg B$, задержку $\mathbf{lat}[B]$ будем считать равной ∞. С каждым отношением отклика $A \rightsquigarrow B$ можно связать *максимальную задержку* $\mathbf{Lat}[A \rightsquigarrow B]$ за все время выполнения:

$$(\mathbf{Lat}[A \rightsquigarrow B] \le k) \equiv \mathbf{G}\big(A \rightarrow (\mathbf{lat}[B] \le k)\big). \qquad (1)$$

Если событие A не наступает ни разу, удобно считать, что $\mathbf{Lat}[A \rightsquigarrow B] = 0$.

Естественно представить задержку обработки запроса P в модели F на рис. 2 как[3]

$$\mathbf{Lat}[(a = P) \rightsquigarrow \mathrm{h.xfer} \cdot (h = P)],$$

т.е. как максимальное число тактов между появлением пакета P на канале a и передачей пакета P на канале h. Однако такое определение не вполне корректно, так как события $(a = P)$ и $\mathrm{h.xfer} \cdot (h = P)$ могут относиться к разным пакетам, имеющим одно и то же значение P. Проблема решается простым изменением модели. Допустим, что исток, посылающий пакеты P, может присвоить одному из них особую пометку, превратив его в пакет P^* (такое поведение реализуется небольшим фрагментом из примитивов xMAS). Помеченный пакет обрабатывается моделью так же, как и обычный пакет P, сохраняя при этом пометку. Искомая задержка для помеченного пакета может быть корректно выражена как

$$\mathbf{Lat}[(a = P^*) \rightsquigarrow \mathrm{h.xfer} \cdot (h = P^*)]. \qquad (2)$$

По построению, модель в каждый момент времени содержит не более одного экземпляра P^*, позволяя однозначно идентифицировать пакет. Пометка присваивается случайно, поэтому каждый пакет P превращается в P^* в некоторых исполнениях модели. Следовательно, запросы P^* и P обладают одинаковой максимальной задержкой. Описанный прием добавления пометки применим к произвольным моделям xMAS.

В дальнейшем нам также потребуется обобщенная форма отношения отклика и связанная с ним задержка:

$$A \rightsquigarrow_E B \equiv \mathbf{G}(A \cdot \mathbf{GF}E \rightarrow \mathbf{F}B).$$

Событие B следует за A, только если в ходе ожидания периодически наступает событие E, называемое *модулем*. Локальная задержка $\mathbf{lat}_E[B]$ определяется как число наступлений события E до ближайшего наступления события B (событие E, наступающее одновременно с B, при этом не учитывается). Максимальная задержка $\mathbf{Lat}[A \rightsquigarrow_E B]$ определяется аналогично (1). Выполнено простое соотношение:

$$\mathbf{Lat}[A \rightsquigarrow B] = \mathbf{Lat}[A \rightsquigarrow_{\mathbf{True}} B].$$

B. Вычисление задержки

Задержка для сложного отношения отклика может быть сведена к вычислению более простых задержек. Например, пусть известно, что для некоторых событий A, B, C

$$\mathbf{Lat}[A \rightsquigarrow B] \le l,\ \mathbf{Lat}[B \rightsquigarrow C] \le k.$$

Тогда верно и то, что между появлением события A и последующим появлением события C проходит не более $l + k$ тактов, т.е.

$$\mathbf{Lat}[\mathrm{A} \rightsquigarrow \mathrm{C}] \le l + k.$$

[3] Выражение "$a = P$" – это сокращение для "$\mathrm{a.irdy} \cdot (\mathrm{a.data} = P)$".

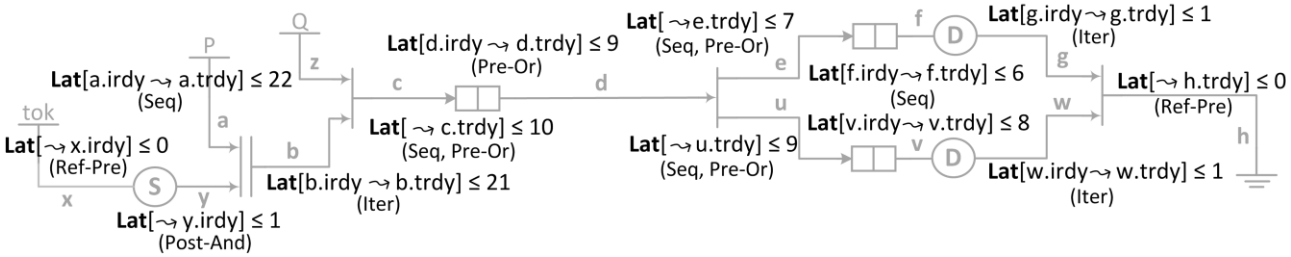

Рис. 3. Локальные задержки в модели коммуникационной фабрики.
Используется сокращение: $\mathbf{Lat}[\rightsquigarrow A] = \mathbf{Lat}[\mathbf{True} \rightsquigarrow A]$

Найденную закономерность можно сформулировать в виде синтаксического правила вывода

$$\frac{\mathbf{Lat}[A \rightsquigarrow B] \le l, \quad \mathbf{Lat}[B \rightsquigarrow C] \le k}{\mathbf{Lat}[A \rightsquigarrow C] \le l + k} \ (\text{Seq})$$

В табл. 1 перечислены правила вывода, подобные (Seq). Символ "$A \rhd B$" обозначает *устойчивость* события A относительно события B:

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

Устойчивость – характерное свойство протокола передачи данных xMAS. Например, $\mathrm{u.irdy} \rhd \mathrm{u.trdy}$ и $\mathrm{u.trdy} \rhd \mathrm{u.irdy}$ для любого канала u.

Теорема 1. Если формула ϕ выводима из гипотез Γ с помощью правил вывода $\mathcal{L}$ (табл. 1), то $\Gamma \vDash \phi$.

Докажем, что в модели F справедлива оценка

$$\mathbf{Lat}[(a = P^*) \rightsquigarrow \mathrm{h.xfer} \cdot (h = P^*)] \le 56.$$

На пути от канала a к каналу h, пакет P^* проходит положения 1-5, показанные на рис. 2. Табл. 2 с каждым положением i связывает выходной канал u_i соответствующего истока или очереди и предикат λ_i, истинный тогда и только тогда, когда P^* находится в положении i. (Выражение $q_i[j]$ определяет значение пакета в j-й позиции, считая от хвоста очереди q_i). Обозначим предусловие $(a = P^*)$ через α и постусловие $\mathrm{h.xfer} \cdot (h = P^*)$ через β.

Пользуясь правилами $\mathcal{L}$, можно вывести

$$\mathbf{Lat}[\alpha \rightsquigarrow \beta] \le \textstyle\sum_i \mathbf{Lat}[\mathrm{u_i.irdy} \rightsquigarrow \mathrm{u_i.trdy}] + (n-1). \quad (3)$$

Локальная задержка $\mathbf{Lat}[\mathrm{u_i.irdy} \rightsquigarrow \mathrm{u_i.trdy}]$ ограничивает время пребывания пакета в положении i по наихудшему случаю, а слагаемое $(n-1)$ учитывает постоянные задержки перехода между положениями.

Таблица 2

Положения пакета P^* в модели на рис. 2

Номер	Положение	Предикат (λ_i)	Канал (u_i)
1	Выход истока	$a = P^*$	a
2	Хвост очереди q_1	$q_1[0] = P^*$	d
3	Голова очереди q_1	$q_1[1] = P^*$	d
4	Хвост очереди q_2	$q_2[0] = P^*$	f
5	Голова очереди q_2	$q_2[1] = P^*$	f

Вывод локальных задержек на каналах схематично показан на рис. 3. Оценки вычисляются последовательно в процессе обхода модели, начиная с каналов h и x, по направлению к каналу a. Привести вывод целиком не представляется возможным из-за нехватки места. Мы рассмотрим в качестве примера вычисление задержки на канале f. Пусть известно, что:

$$\mathbf{Lat}[\mathrm{g.irdy} \rightsquigarrow \mathrm{g.trdy}] \le 1. \quad (4)$$

Канал g является выходом ограниченной задержки с входом f. Из свойств примитива следует, что

$$\mathbf{Lat}[\mathrm{f.irdy} \rightsquigarrow (\mathrm{cnt} = 0)] \le 5, \quad (5)$$

то есть при наличии сигнала f.irdy внутренний счетчик задержки убывает до 0 не более чем за $d = 5$ тактов. Принимая неравенства (4) и (5) в качестве гипотез, можно показать, что

$$\mathbf{Lat}[\mathrm{f.irdy} \rightsquigarrow \mathrm{f.trdy}] \le 6.$$

Пошаговый вывод, приведенный в табл. 3, опирается только на семантику примитива и свойства протокола xMAS и может быть обобщен в виде вспомогательного *правила распространения* для примитива ограниченной задержки с параметром d, входом i и выходом o:

$$\frac{\mathbf{Lat}[\mathrm{o.irdy} \rightsquigarrow \mathrm{o.trdy}] \le k}{\mathbf{Lat}[\mathrm{i.irdy} \rightsquigarrow \mathrm{i.trdy}] \le k + d}$$

Правило опирается на истинность гипотезы (5) и ряд других свойств примитива и протокола, указанных в табл. 3 (например, $\mathrm{g.irdy} \cdot \mathrm{g.trdy} \rightarrow \mathrm{f.trdy}$).

С помощью теоремы 1 можно вывести аналогичные правила для всех примитивов xMAS. Например, для слияния с входом a и выходом o

$$\frac{\mathbf{Lat}[\mathrm{o.irdy} \rightsquigarrow \mathrm{o.trdy}] \le k}{\mathbf{Lat}[\mathrm{a.irdy} \rightsquigarrow \mathrm{a.trdy}] \le k + L_a + k \cdot L_a}$$

При этом предполагается истинным неравенство

$$\mathbf{Lat}[\mathrm{a.irdy} \rightsquigarrow_{\mathrm{o.xfer}} \sigma_a] \le L_a.$$

Задержка арбитража L_a равна максимальному числу раундов арбитража до получения входом a наивысшего приоритета. Понятие "наивысшего приоритета" зависит от выбранного алгоритма арбитража и задается *предикатом выбора* σ_a, удовлетворяющего требованиям

$$(\mathrm{a.irdy} \cdot \sigma_a) \rhd \mathrm{o.xfer},$$
$$\mathrm{a.trdy} = \mathrm{a.irdy} \cdot \sigma_a \cdot \mathrm{o.trdy},$$

т.е. истинность σ_a гарантирует, что пакет на канале a выиграет следующий раунд. Предикат σ_a и задержка L_a могут быть определены для большинства справедливых алгоритмов арбитража. В случае кругового алгоритма состояние арбитра задается целочисленной переменной s, значение которой равно номеру текущего выбранного входа. Тогда предикат выбора σ_a можно положить равным $(\mathrm{s} = i_a)$, где i_a – номер, назначенный входу a; задержка арбитража L_a равна общему числу входов минус один.

Таблица 3

Вывод оценки на канале f в модели F

	Пусть
γ_1	**Lat**[g. irdy ⇝ g. trdy] ≤ 1
γ_2	**Lat**[f. irdy ⇝ (cnt = 0)] ≤ 5
	Тогда
1	**Lat**[f. irdy ⇝ f. irdy · (cnt = 0)] ≤ 5
(Persist)	γ_2, f. irdy ⊳ (cnt = 0)
2	**Lat**[f. irdy ⇝ g. irdy] ≤ 5
	1, g. irdy = f. irdy · (cnt = 0)
3	**Lat**[g. irdy ⇝ g. irdy · g. trdy] ≤ 1
(Persist)	γ_1, g. irdy ⊳ g. trdy
4	**Lat**[f. irdy ⇝ g. irdy · g. trdy] ≤ 6
(Seq)	2, 3
5	**Lat**[f. irdy ⇝ f. trdy] ≤ 6
(Post-Or)	4, g. irdy · g. trdy → f. trdy

Подставляя оценки для каналов a, d и f (рис. 3) в формулу (3), получаем оценку задержки (2):

$$\mathbf{Lat}[\alpha \rightsquigarrow \beta] \le 22 + 2 \cdot 9 + 2 \cdot 6 + 4 = 56. \qquad (6)$$

Описанный метод анализа задержек применим к широкому классу микроархитектурных моделей. При отсутствии структурных циклов все локальные задержки находятся последовательным применением соответствующих правил распространения. Оценка для многошаговой транзакции получается суммированием локальных оценок по всей ее траектории.

Для моделей со структурными циклами (например, виртуальный канал с кредитной логикой [2]) необходимо учитывать множество достижимых состояний системы. Мы планируем рассмотреть циклические модели более подробно в дальнейшем. Однако нам представляется, что предложенный метод анализа применим и в этом случае, возможно с некоторыми модификациями.

IV. Ранжирующие функции

В разделе III мы воспользовались правилами вывода $\mathcal{L}$ для построения оценки на задержку обработки пакета. Доказательство опирается на ряд предположений о свойствах примитивов и протокола передачи данных. На практике принятые гипотезы могут оказаться ложными (например, из-за изменений в семантике примитива), а в ходе вывода могут быть допущены ошибки. Для проверки правильности оценки мы воспользуемся средствами верификации моделей и методом ранжирующих функций.

Для двух произвольных событий A и B определим *интервал от A до B* как

$$A{:}B = A + \mathbf{pre}\big((A{:}B) \cdot \neg B\big).$$

Оператор **pre** возвращает значение своего аргумента на предыдущем такте (**False** в начальный момент времени). Выражение (A: B) становится истинным при каждом наступлении события A и остается таковым до первого наступления события B.

Целая неотрицательная величина ϕ называется *ранжирующей функцией* для отношения отклика $A \rightsquigarrow_E$ B, если

$$\mathbf{G}\big((A{:}B) \cdot \neg B \rightarrow (\phi^+ \le \phi - [E])\big), \qquad (7)$$

где $[E]$ равно 1 в момент наступления E и 0 в остальных случаях. Условие $(A{:}B) \cdot \neg B$ означает, что событие A уже наступило, а событие B – еще нет. В этой ситуации условие (7) требует невозрастания ϕ при $E = \mathbf{False}$ и строгого убывания ϕ при $E = \mathbf{True}$. Содержательно величину ϕ можно интерпретировать как "меру расстояния" в единицах E от текущего момента времени до наступления события B. Примем обозначение $\mathbf{rk}[A \rightsquigarrow_E \mathrm{B}] \doteq \phi$, где символ "$\doteq$" читается как "можно положить равным".

Теорема 2. Пусть $\mathbf{rk}[A \rightsquigarrow_E \mathrm{B}] \doteq \phi$ и выполнено $\mathbf{G}(A \rightarrow (\phi \le l))$. Тогда $\mathbf{Lat}[A \rightsquigarrow_E \mathrm{B}] \le l$.

Согласно теореме 2, для обоснования оценки l на задержку достаточно построить ранжирующую функцию ϕ и доказать верхнюю границу $\mathbf{G}(A \rightarrow (\phi \le l))$. На практике часто удается найти простые ранжирующие функции и выполнить проверку условий теоремы 2 средствами одношаговой индукции.

Аналогично задержке вычисление сложной ранжирующей функции может быть сведено к более простым случаям. Однако при комбинировании ранжирующих функций для разных отношений отклика возникает техническая трудность, связанная с неопределенностью поведения функции ϕ, удовлетворяющей условию (7), вне интервала $A{:}B$. Пусть, например,

$$\mathbf{rk}[A \rightsquigarrow B] \doteq \phi, \quad \mathbf{rk}[B \rightsquigarrow C] \doteq \psi.$$

По аналогии с правилом (Seq) для задержки, естественно было бы определить ранжирующую функцию для $A \rightsquigarrow C$ как $\phi + \psi$. Однако поведение функции ψ может быть произвольным между наступлением события A и наступлением события B (аналогично, для функции ϕ на интервале от B до C). Это не позволяет гарантировать убывания суммы $\phi + \psi$ на всем интервале $A{:}C$. В таких случаях значения функций доопределяются с помощью специальных операторов $\downarrow_X$ и $\uparrow_X^M$.

Для целочисленной величины ϕ, условия X и постоянной M определим

$$\phi \downarrow_X \equiv \phi \cdot [\neg X], \quad \phi \uparrow_X^M \equiv \phi \cdot [\neg X] + M \cdot [X].$$

Используя доопределяющие операторы, можно сформулировать правило:

$$\frac{\mathbf{rk}[A \rightsquigarrow_E B] \doteq \phi,\quad \mathbf{rk}[B \rightsquigarrow_E C] \doteq \psi,\quad \mathbf{G}(B \to (\psi \le M))}{\mathbf{rk}[A \rightsquigarrow_E C] \doteq \phi \downarrow_{B:C} + \psi \uparrow^{M}_{\neg(B:C)}} \text{(Rk-Seq)}$$

Аналогично (Rk-Seq), каждому правилу вывода для задержек можно поставить в соответствие правило для ранжирующих функций. Роль гипотез при выводе выполняют базовые ранжирующие функции, описывающие поведение примитивов. Например, для примитива задержки с входом i, выходом o и параметром d справедливо равенство

$$\mathbf{rk}[\mathrm{i.irdy} \rightsquigarrow (\mathrm{cnt} = 0)] \doteq \mathrm{cnt}.$$

Повторяя вывод, показанный на рис. 3, можно найти $\mathbf{rk}[\mathrm{f.irdy} \rightsquigarrow \mathrm{f.trdy}]$, $\mathbf{rk}[\mathrm{d.irdy} \rightsquigarrow \mathrm{d.trdy}]$ и $\mathbf{rk}[\mathrm{a.irdy} \rightsquigarrow \mathrm{a.trdy}]$. Из них по формуле аналогичной (3) получается ранжирующая функция для полного отношения отклика: $\mathbf{rk}[(a = P^*) \rightsquigarrow \mathrm{h.xfer} \cdot (h = P^*)]$. Далее с помощью одношаговой индукции можно установить свойство (7) и верхнюю границу 56, тем самым верифицировав оценку (6).

V. Результаты экспериментов

Метод анализа, описанный в разделах III-IV, тестировался на небольших примерах. В табл. 4 сравниваются оценки на задержку обработки пакета P в модели F на рис. 2, полученные разными способами. Напомним, что общая задержка (от a до h) ограничена задержками на каналах f, d и a по формуле (3).

Метод ограниченной верификации моделей (BMC) исследует все исполнения из начального состояния в течение ограниченного числа тактов и позволяет получить нижнюю оценку задержки. k-индукция рассматривает исполнения модели из произвольного состояния (возможно ограниченного дополнительными инвариантами [2]) и дает консервативную оценку сверху.

Таблица 4

Сравнение оценок задержки для модели на рис. 2

Задержка	BMC	k-индукция	Вывод по правилам $\mathcal{L}$
f	6	6	6
d	7	8	8
a	13	17	21
$a \rightsquigarrow$ h	32	34	56

Оценки для BMC и k-индукции оказываются приблизительно равными. Небольшое расхождение обусловлено случайной синхронизацией состояний различных примитивов (например, внутренних счетчиков задержек и формирователей, текущих значений приоритетов в алгоритмах арбитража и т.д.). Такие эффекты крайне тяжело учесть, не прибегая к полному перебору состояний модели. Более интересны различия между двумя последними столбцами табл. 4. Полная задержка, полученная по правилам $\mathcal{L}$, оказывается почти вдвое больше оценки методом k-индукции. Расхождение можно сократить, повышая точность правил распространения, но полного совпадения достигнуть не удастся. Это связано с тем, что оценка в 56 тактов получена по формуле (3), оценивающей время пребывания в каждой точке траектории по наихудшему случаю. В реальных исполнениях модели наихудшее значение задержки может возникать в каждой отдельной точке траектории, но не во всех точках одновременно.

Использование ранжирующих функций должно давать значительный выигрыш по производительности для сложных примеров. Например, для моделей с числом очередей порядка 10-100, оценка BMC часто далека от точной, а k-шаговая индукция не завершается за разумное время. В дальнейшем мы планируем проверить эффективность метода ранжирующих функций на больших моделях экспериментально. Для простых моделей большинство алгоритмов верификации позволяет получить результат за короткое время (порядка нескольких секунд), и сравнение их быстродействия не представляет интереса.

VI. Заключение

Мы разработали метод анализа задержек в микроархитектурных моделях коммуникационных фабрик. Метод основан на последовательном использовании элементарных свойств задержки для вывода все более сложных оценок из простых гипотез, описывающих поведение отдельных примитивов и свойства протокола xMAS. Итоговая оценка консервативно предполагает наихудшую задержку в каждой точке траектории. Нами предложен способ построения ранжирующих функций, позволяющий проводить быструю верификацию построенных оценок.

В дальнейшем мы планируем повысить точность метода и обобщить его для моделей произвольного вида. Нам представляется, что рассмотренный подход к построению оценок задержки и ранжирующих функций применим для анализа широкого класса микроархитектурных описаний.

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 // in 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 // in proc. CAV, LNCS. 2003. V. 2725. P. 1-13.

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

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

[7] Chawdhary A., Cook B., Gulwani S., Sagiv M., Yang H. Ranking Abstractions // in proc. ESOP, LNCS. 2008. V. 4960. P. 81-92.