ОЦЕНКА ПРИМЕНИМОСТИ ПРОЦЕССОРНЫХ ЯДЕР KMX32 И RISC-V В СЕТЕВОМ ПРОЦЕССОРНОМ УСТРОЙСТВЕ[5]

С.О. Беззубцев, Д.Ю. Волканов, Ш.Р. Жайлауова, Ю.А. Скобцова, Р.Л. Смелянский, А.А. Сухова (Москва)

В конце прошлого столетия в качестве основного устройства обработки и передачи трафика в коммутаторах использовались либо микропроцессоры общего назначения, либо специализированные интегральные схемы. Каждый из этих вариантов обладает своими преимуществами и недостатками. В начале 2000-х появились устройства, выполняющие функции обработки и передачи трафика, но занимающие промежуточное место между микропроцессорами общего назначения и специализированными интегральными схемами - сетевые процессорные устройства (СПУ). В отличие от микропроцессоров общего назначения, СПУ ориентировано на эффективное выполнение задач, возникающих в коммутаторах, но область применения их более широка, чем в специализированных схемах. С появлением технологии программно-конфигурируемых сетей [3] интерес к СПУ возрос, и связано это, прежде всего, с возможностью программировать СПУ согласно возникающим потребностям.

Процесс обработки пакетов можно разделить на несколько последовательных этапов [2]: выделение заголовка, классификация и модификация, что естественно позволяет нам представить СПУ в виде конвейера вычислительных узлов, каждый из которых выполняет свои задачи. Специализацию каждого узла или группы узлов можно обеспечить программным или аппаратным способом. В первое случае, все узлы будут однородными по своим вычислительным способностям, но программно настроены на выполнение различных задач. Во втором случае, узлы, принадлежащие разным стадиям конвейера будут аппаратно отличаться друг от друга. В этой статье мы уделим внимание первому случаю и исследуем два процессорных ядра: KMX32[6] и RISC-V [5] в качестве вычислительных узлов конвейера. Кроме конвейерной организации СПУ, существует следующая: обработку одного пакета от начала и до конца выполняет один вычислительный узел, следовательно, СПУ будет состоять из набора узлов, работающих синхронно или асинхронно.

Общее описание архитектуры СПУ и постановки задачи приведено в следующем разделе.

Постановка задачи

В основе рассматриваемой архитектуры СПУ (рис. 1) лежит последовательный конвейер, стадии которого функционально различны и представлены 5 типами:

- стадия типа Receive – прием пакета, отделение заголовка и сохранение тела пакета в памяти;
- стадия типа Parse – разбор заголовка пришедшего пакета;
- стадия типа Lookup – классификация пакета;
- стадия типа Resolve – изменение заголовка, направление пакета на выходной порт;
- стадия типа Replicate – перенос обработанного пакета в буферы коммутационной матрицы, репликация пакетов, «разворот» на начало конвейера.

[5] Работа выполнена при частичной поддержке РФФИ, грант № 19-07-01076

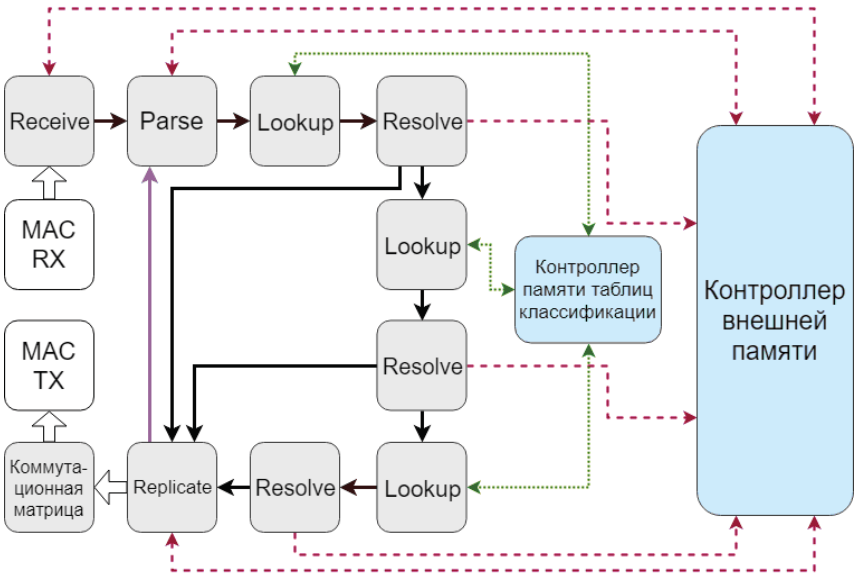

Рис. 1. Структура тракта обработки данных предложенной архитектуры СПУ и схема каналов связи с контроллерами памяти.

В каждой стадии могут использоваться процессорные ядра KMX32 или RISC-V.

Микроконтрольное ядро KMX32 [6] разрядностью 32 бита является одним из представителей семейства микропроцессоров КРОЛИК на базе оригинального RISC ядра компании KM211. Перечислим некоторые особенности данной RISC-архитектуры:

- 3-ступенчатый конвейер обработки команд;
- исполнение большинства команд за 1 такт;
- аппаратный умножитель 16 x 16 бит и аппаратный делитель;
- аппаратная организация циклов и потоковый режим исполнения команд;
- наличие аппаратного или аппаратно-программного режимов стека контекстов;
- наличие двух режимов энергосбережения: WAIT и STOP.

Архитектура RISC-V [5] - это свободная архитектура команд, в основе которой также лежит концепция RISC. Модульность данной архитектуры заключается в том, что имеется обязательное для реализации ядро команд, которое при необходимости можно наращивать, используя расширения, соответствующие конкретным приложениям интегральных схем.

Базовые инструкции, входящие в ядро набора команд архитектуры RISC-V, выровнены и имеют длину в 32 бита. В RISC-V отсутствует поддержка целочисленных проверок переполнения. В базовый набор инструкций входят только следующие целочисленные инструкции: операции сдвига, сложение, вычитание, операции сравнения и логические операции. Несмотря на то, что базовые инструкции составляют 4 байта, RISC-V также поддерживает инструкции переменной длины.

Таким образом, в работе исследуется следующая задача. Дана архитектура конвейерного СПУ со стадиями пяти типов, даны два варианта процессорных ядер (KMX32, RISC-V) для использования в стадиях конвейера СПУ. Требуется определить производительности стадий конвейера, а также сравнить общую производительность архитектур СПУ для каждого из процессорных ядер.

Для решения это задачи была разработана поведенческая имитационная модель СПУ, которая описана в следующем разделе.

Имитационная модель СПУ

Поведенческая имитационная модель СПУ реализована на языках C/C++ с использованием открытой C++ библиотеки SystemC [4].

Предлагаемая модель работы СПУ выполняет следующие действия:

- передача поступившего пакета сетевому процессору;
- запись поступившего пакета в локальную память;
- обработка заголовка;
- в зависимости от результата обработки принятие решения о дальнейшей отправке пакета на соответствующие порты коммутатора или дополнительной обработке пакета;
- проверка работы сетевого процессора.

Базовым элементом в SystemC является модуль, включающий в себя процессы и другие модули. Модель в SystemC состоит из нескольких модулей, которые общаются друг с другом через порты. Модель состоит из трех основных модулей (блоков): Provider, DUT и Consumer. Каждый блок оснащён портами, с помощью которых происходит взаимодействие между блоками. Коммуникация между блоками осуществляется с помощью библиотеки SystemC TLM. Блоки соединены друг с другом однонаправленными каналами из библиотеки TLM.

Все три компонента связаны между собой каналами, организованными по принципу FIFO (Рис. 2).

Рис. 2. Основные компоненты модели СПУ.

Поставщик (Provider) передает пакеты на обработку СПУ (DUT), имитируя поступление пакетов на порты коммутатора. СПУ получив пакет, начинает его обработку. Возможно несколько вариантов исхода обработки пакета. Примеры таких исходов: на каком-либо этапе обработки пакет может быть сброшен; таблица коммутатора не содержит номер порта, на который поступил пакет - в этом случае принимается решение о широковещательной передаче пакета и обучении коммутатора, то есть о внесении номера порта в таблицу; наконец, отправка пакета на необходимые порты и др. Потребитель (Consumer) проверяет корректность обработки пакетов сетевым процессором: ему передается информация о том, на каком этапе завершилась обработка пакета.

На Рис. 3 представлена схема связей между основными объектами СПУ.

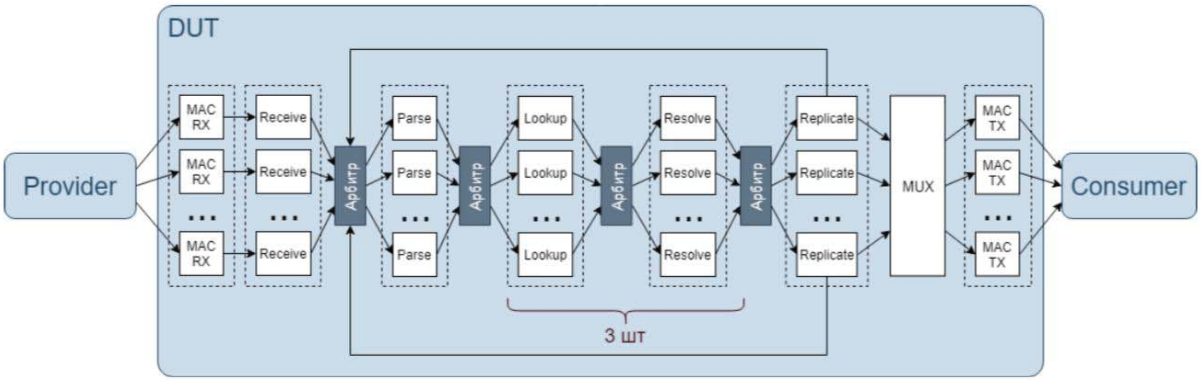

Рис. 3. Схема связей между объектами сетевого процессора

Модель СПУ включает в себя блоки MAC RX и MAC TX (ресиверы и трансмиттеры интерфейса Ethernet, соответственно), контроллеры блоков памяти и конвейер обработки пакетов (На рис. 2 контроллеры памяти не указаны). Внутренняя память, то есть память на кристалле, предназначена для хранения контекстов пакетов, таблиц классификации. Внешняя память, то есть память вне кристалла, предназначена для хранения тел пакетов.

При прохождении конвейера с пакетом ассоциируется контекст, который далее передается блоками друг другу и модифицируется ими. Таким образом, фактически по конвейеру перемещается только контекст пакета. Контекст состоит из заголовка пакета и/или сопутствующих метаданных (как стандартных специфицированных, так и заданных разработчиком). Состав

контекста, то есть наборы полей входных и выходных метаданных, для различных блоков отличается. Любая копия пакета обрабатывается как отдельный пакет со своим собственным контекстом. При программировании конвейера обработки данных необходимо учитывать различия в контекстах между стадиями.

Стадии конвейера состоят из наборов процессорных ядер, каждое из которых обрабатывает один пакет. Таким образом, одна стадия может параллельно работать с несколькими пакетами. Поскольку предполагается использование процессорных ядер общего назначения, стадии имеют высокую гибкость программирования.

В модели используются динамические библиотеки, содержащие реализацию всех ячеек типов Parse, Lookup и Resolve для каждого сценария. При запуске модели указывается, какой сценарий будет промоделирован. Выбор сценария осуществляется выбором реализаций ячеек типов Parse, Lookup и Resolve из динамической библиотеки. Также во время запуска в качестве опций задаются следующие параметры: период тактового сигнала, количество тактов, в течение которого происходит моделирование, набор задержек для каждой ячейки. Если данные параметры при запуске не указываются, то при моделировании используются значения этих параметров по умолчанию.

Экспериментальное исследование

В работе [1] подробно описан набор сценариев обработки пакетов коммутатором, которому должна удовлетворять предлагаемая модель СПУ. Эти сценарии соответствуют сценариям обработки пакетов в сетях регионального уровня. Эти сценарии разбиты на пять групп:

- сценарии работы классического L2-коммутатора с обучением;
- сценарии работы L2/L3 коммутатора;
- сценарии работы сетевых приложений на ПКС-коммутаторе;
- сценарии агрегирования, очередизации и перенаправления трафика на ПКС-коммутаторе;
- сценарии обработки трафика в MPLS сетях.

Сравнительный анализ применимости ядер KMX32 и RISC-V в качестве вычислительных узлов в рассматриваемой модели СПУ в данной работе был произведен на основе этих сценариев.

Для каждого сценария проводилось измерение времени выполнения стадий конвейера Receive, Parse, Lookup, Resolve и Replicate и расчет пропускной способности каждой из стадий конвейера по следующей методике:

1) Измерение времени работы функций, вызываемых на каждой из стадий, для каждого сценария на эмуляторе процессора KMX32 или эмуляторе RISC-V.

2) Запуск имитационной модели СПУ с каждым из сценариев. В качестве исполняемых стадиями функций на вход модели подаются адреса из динамической библиотеки функций сценария. В качестве задержек работы каждой из стадий указываются значения, полученные на шаге 1.

3) Сохранение итогового времени работы в тактах на каждой из стадий с учётом моделируемых задержек.

4) Сохранение числа пакетов и времени их обработки в тактах конвейером целиком (с учетом блоков MAC).

Результаты измерений времени и пропускной способности каждой стадии представлены в таблицах 1 — 4. Сравнение пропускной способности каждой стадии для ядер KMX32 и RISC-V показывает, что время выполнения стадий Receive, Lookup и Replicate для обоих архитектур находятся примерно в одном диапазоне. Что касается стадий Parse и Resolve, то результат при использовании ядер RISC-V в 1,5-3 раза лучше, чем при использовании ядер KMX32.

Таблица 1.Результаты оценки производительности стадий конвейера в случае ядер KMX32

Сценарии	Receive	Parse	Lookup 1	Resolve 1	Lookup 2	Resolve 2	Lookup 3	Resolve 3	Replicate
Multicast	327	2233	2509	2175	7438	5916	*	*	3919
B2C-AR	327	2195	2466	2146	2100	2347	*	*	3919
B2C-DR	327	2283	2932	2185	2812	2267	4774	6608	3919
P2P	327	2254	2544	2271	10179	3005	*	*	3919
Multipoint	327	2318	7450	10199	2579	2238	2825	2272	3919
LACP/LLDP/QoS	327	2283	2345	2390	*	*	*	*	3919
LAG	327	4298	2729	2433	2596	2433	8178	2551	3919
Mirror	327	2167	2153	2375	*	*	*	*	3919
Inband	327	2521	3138	4389	2386	4479	6201	2558	3919
MPLS ingress	327	2788	2314	2419	2691	2348	2255	4368	3919
MPLS VPN ingress	327	2786	2280	2716	2930	5855	2219	2220	3919
MPLS-P	327	2353	2227	2354	2494	4085	*	*	3919
MPLS egress	327	2713	2214	2345	2143	3955	*	*	3919
MPLS VPN egress	327	2517	2298	2677	2211	3576	2229	2247	3919

Таблица 2.Результаты оценки производительности стадий конвейера в случае ядер RISC-V

Сценарии	Receive	Parse	Lookup 1	Resolve 1	Lookup 2	Resolve 2	Lookup 3	Resolve 3	Replicate
Multicast	464	908	8211	998	8694	3482	*	*	4083
B2C-AR	457	876	2082	651	1881	1855	*	*	3597
B2C-DR	424	1561	2264	794	2944	1502	3030	1487	4887
P2P	482	950	3056	839	2437	3147	*	*	3822
Multipoint	494	1123	4562	2306	2507	893	3739	1256	3996
LACP/LLDP/QoS	446	1598	1939	841	*	*	*	*	4564
LAG	405	522	2712	769	1901	746	2317	2717	4415
Mirror	410	1230	3095	891	*	*	*	*	3172
Inband	535	1886	3780	814	3075	575	3340	899	3265
MPLS ingress	436	2304	3704	1666	3637	1409	3197	1549	3880
MPLS VPN ingress	479	2312	2583	1578	4037	2596	2083	1071	4374
MPLS-P	407	1170	3571	1542	2292	1884	*	*	4631
MPLS egress	453	1490	2461	1469	2055	1984	*	*	3515
MPLS VPN egress	374	1743	2676	1584	1902	1499	3237	1651	3046

Таблица 3. Результаты оценки пропускной способности стадий конвейера в Гбит/с в случае ядер KMX32

Сценарии	Receive	Parse	Lookup 1	Resolve 1	Lookup 2	Resolve 2	Lookup 3	Resolve 3	Replicate
Multicast	3,18	8,60	7,42	8,83	2,57	3,25	*	*	3,29
B2C-AR	3,13	8,75	7,45	8,95	8,70	8,18	*	*	3,40
B2C-DR	3,13	8,41	6,30	8,79	6,64	8,47	3,92	2,91	3,13
P2P	3,13	8,52	7,18	8,45	1,87	6,39	*	*	3,43
Multipoint	3,13	8,28	2,54	1,88	7,20	8,58	6,48	8,45	0,11
LACP/LLDP/QoS	2,88	8,41	7,58	7,81	*	*	*	*	3,18
LAG	3,18	8,52	4,36	7,04	7,03	7,89	2,31	7,53	3,50
Mirror	2,94	3,77	8,44	8,08	*	*	*	*	3,47
Inband	3,18	7,62	5,88	4,37	7,73	4,29	3,07	7,51	3,29
MPLS ingress	3,13	6,89	7,74	4,40	6,79	8,18	8,03	8,47	3,31
MPLS VPN ingress	3,13	6,89	7,84	7,07	6,26	3,28	8,22	8,61	3,49
MPLS-P	2,69	8,16	8,25	8,16	7,31	4,70	*	*	0,51
MPLS egress	2,92	7,08	8,18	8,19	8,76	4,85	*	*	0,34
MPLS VPN egress	2,92	7,63	8,00	7,17	8,26	5,37	8,55	8,54	3,46

Таблица 4. Результаты оценки пропускной способности стадий конвейера в Мбит/с в случае ядер RISC-V

Сценарии	Receive	Parse	Lookup 1	Resolve 1	Lookup 2	Resolve 2	Lookup 3	Resolve 3	Replicate
Multicast	4,13	21,14	2,33	19,23	2,2	5,51	*	*	4,7
B2C-AR	2,19	21,91	9,22	29,49	10,2	10,35	*	*	5,33
B2C-DR	4,52	12,29	8,48	24,18	6,52	12,78	6,33	12,91	3,92
P2P	3,98	20,21	6,28	22,88	7,87	3,10	*	*	5,02
Multipoint	3,88	17,09	4,2	8,32	7,62	21,5	5,135	15,28	4,8
LACP/LLDP/QoS	4,3	12,01	9,9	22,82	*	*	*	*	4,2
LAG	4,74	36,78	7,07	24,96	10,09	25,73	8,28	7,06	4,34
Mirror	4,67	15,6	6,2	21,5	*	*	*	*	6,05
Inband	3,58	10,18	5,07	23,58	6,24	33,39	5,74	21,35	5,88
MPLS ingress	4,4	8,33	5,18	11,52	5,27	13,62	6	12,39	4,94
MPLS VPN ingress	4	8,3	7,43	12,16	4,75	7,39	9,21	17,92	4,38
MPLS-P	4,71	16,41	5,37	12,45	8,37	10,19	*	*	4,14
MPLS egress	4,23	12,88	7,8	13,07	9,34	9,67	*	*	5,46
MPLS VPN egress	5,13	11,01	7,17	12,12	10,09	12,8	5,93	11,62	6,3

Таблица 5. Сравнение пропускной способности конвейера в Гбит/с для ядер KMX32 и RISC-V

Сценарии	Ядра RISC-V	Ядра KMX32
Multicast	2,76	2,91
B2C-AR	3,05	3,6
B2C-DR	4,8	3,6
P2P	3,8	3,5
Multipoint	4,7	0,6
LACP/LLDP/QoS	4,7	3,14
LAG	5,6	2,8
Mirror	5,2	3,2
Inband	4,7	3,5
MPLS ingress	5,07	3,7
MPLS VPN ingress	4,7	3,6
MPLS-P	4,7	0,89
MPLS egress	4,8	0,74
MPLS VPN egress	5,9	3,5

Заключение

Эффективность ядер RISC-V по сравнению с ядрами KMX32 достигается значительным образом только на стадиях Parse и Resolve. Несмотря на то, что взаимодействие с памятью вносит наибольший вес в длительность обработки пакета конвейером ускорение в ядрах RISC-V довольно существенное. Таким образом, предпочтительней использование ядер архитектуры RISC-V в качестве вычислительных узлов СПУ, чем ядер KMX32.

Литература

1. Беззубцев С.О., Волканов Д.Ю., Жайлауова Ш.Р., Мирошник В.А., Скобцова Ю.А., Смелянский Р.Л.. Об одном подходе к построению сетевого процессорного устройства. // Моделирование и анализ информационных систем. - 2019. - Т. 26, № 1. - С. 39–62.
2. Kornaros G. Multi-core embedded systems // Boca Raton, FL: CRC Press. 2014
3. Смелянский Р.Л. Программно-конфигурируемые сети // Открытые системы. СУБД. - 2012. - № 9. - С. 15-26.
4. Accelera Standarts: SystemC // http://www.accellera.org/downloads/standards/systemc
5. RISC-V architecture// http://www.riscv.org
6. Семейство микроконтроллеров KMX32 // http://www.km211.ru