СИСТЕМЫ И СРЕДСТВА ИНФОРМАТИКИ

Научный журнал Российской академии наук
(издается под руководством Отделения нанотехнологий и информационных технологий РАН)

Издается с 1989 года
Журнал выходит ежеквартально

Учредитель:
Федеральный исследовательский центр «Информатика и управление» Российской академии наук

РЕДАКЦИОННЫЙ СОВЕТ

РЕДАКЦИОННАЯ КОЛЛЕГИЯ

Редакция

Журнал включен в базу данных Russian Science Citation Index (RSCI), интегрированную с Web of Science
Журнал входит в систему Российского индекса научного цитирования (РИНЦ)
Журнал включен в базу данных CrossRef (систему DOI — Digital Object Identifier), в базу данных Ulrich's periodicals directory и в информационную систему «Общероссийский математический портал Math-Net.Ru»

Журнал реферируется в «Реферативном журнале» ВИНИТИ и в системе Google Scholar

Журнал включен в сформированный Минобрнауки России Перечень рецензируемых научных изданий, в которых должны быть опубликованы основные научные результаты диссертаций на соискание ученой степени кандидата наук, на соискание ученой степени доктора наук

http://www.ipiran.ru/journal/collected

СИСТЕМЫ И СРЕДСТВА ИНФОРМАТИКИ

Том 31 № 3 Год 2021

СОДЕРЖАНИЕ

СИСТЕМЫ И СРЕДСТВА ИНФОРМАТИКИ

Том 31 № 3 Год 2021

СОДЕРЖАНИЕ

АППАРАТНАЯ РЕАЛИЗАЦИЯ РЕКУРРЕНТНОГО ОБРАБОТЧИКА СИГНАЛОВ*

***Ю. А. Степченков*[1], *Н. В. Морозов*[2], *Ю. Г. Дьяченко*[3], *Д. В. Хилько*[4]**

Аннотация: Представлены результаты аппаратной реализации гибридной многоядерной архитектуры рекуррентного сигнального процессора (ГМАРСП) в виде VHDL-модели уровня регистровых передач и ее апробации в виде макетного образца на отладочной плате с программируемой логической интегральной схемой (ПЛИС) Intel Arria10. Гибридная многоядерная архитектура рекуррентного сигнального процессора состоит из ведущего фон-неймановского процессора, реализующего управляющий уровень архитектуры, и потокового процессора с четырьмя вычислительными секциями на операционном уровне архитектуры. Аппаратная модель ГМАРСП представляет собой совокупность программной или аппаратной реализации управляющего процессора (УП) и VHDL-модели операционного уровня ГМАРСП. Программная реализация УП предоставляется системой Quartus автоматизированного проектирования цифровых СБИС на ПЛИС фирмы Intel. Аппаратную реализацию УП в виде двухъядерного процессора Cortex-A9 обеспечивает ПЛИС на отладочной плате.

Ключевые слова: рекуррентный сигнальный процессор; гибридная многоядерная архитектура; VHDL-модель; ПЛИС

DOI: 10.14357/08696527210310

1 Введение

Гибридная многоядерная архитектура рекуррентного сигнального процессора [1] — это альтернатива традиционной вычислительной архитектуре. Она ориентирована на решение задач цифровой обработки сигналов (ЦОС) и относится к классу потоковых архитектур, но свободна от недостатков, присущих потоковым архитектурам массового параллелизма на базе ассоциативной памяти:

– высокой аппаратной сложности и времени сравнения тегированных маркеров;

– неэффективного исполнения последовательных участков кода;

*Исследование выполнено при поддержке Российского научного фонда (проект 19-11-00334).

[1]Федеральный исследовательский центр «Информатика и управление» Российской академии наук, YStepchenkov@ipiran.ru

[2]Федеральный исследовательский центр «Информатика и управление» Российской академии наук, NMorozov@ipiran.ru

[3]Федеральный исследовательский центр «Информатика и управление» Российской академии наук, diaura@mail.ru

[4]Федеральный исследовательский центр «Информатика и управление» Российской академии наук, dhilko@yandex.ru

– чересчур большого числа команд, требующегося для выполнения программы, по отношению к процессорам традиционной архитектуры.

– использования сложной, медленной и энергоемкой ассоциативной памяти в качестве устройства сопоставления;

– большого размера теговых полей, превышающего в десятки раз размер собственно обрабатываемых данных.

Учет этих и других факторов потребовал особого подхода к внедрению потоковой парадигмы в область ЦОС. ГМАРСП — это компромиссное решение, обеспечивающее совместимость с существующими вычислительными и аппаратными средами.

Основным средством разработки и отладки капсул для ГМАРСП служит имитационная модель ГМАРСП [1]. Она учитывает особенности архитектурной организации ГМАРСП и позволяет разработчику капсулы проследить за продвижением данных по этапам работы алгоритма и их распределением по параллельным вычислительным ресурсам. Имитационная модель ГМАРСП облегчает процесс капсульного программирования. Однако она не учитывает особенности аппаратной реализации архитектуры и не может быть использована в качестве исходного алгоритмического описания ГМАРСП для ее синтеза в виде аппаратуры с помощью промышленных средств автоматизированного проектирования.

Поэтому разработка аппаратной модели ГМАРСП на уровне регистровых передач на языке VHDL, обеспечивающей адекватность аппаратной реализации ГМАРСП ее архитектуре и позволяющей оценить характеристики ГМАРСП на отладочной плате с ПЛИС, становится актуальной задачей. Данная статья описывает особенности реализации аппаратной модели ГМАРСП в виде макетного образца на отладочной плате HAN Pilot Platform с ПЛИС Intel Arria10 SoC 10AS066K3F40E2SG [2], ее технические характеристики и требуемые аппаратные ресурсы в терминах структурных единиц, имеющихся в составе ПЛИС.

2 Особенности аппаратной реализации ГМАРСП

ГМАРСП имеет следующие особенности аппаратной реализации, преодолевающие недостатки известных потоковых архитектур.

1. Работа ассоциативного запоминающего устройства (АЗУ) основана на сравнении теговых полей входного или искомого операнда с теговыми полями всех операндов в АЗУ. Адрес ячейки АЗУ, в которой теговые поля совпали, выбирается для выполнения операции записи/чтения. Это существенно усложняет АЗУ по сравнению с адресуемой памятью, замедляет его и увеличивает энергопотребление. Вместо большого и медленного массива ассоциативной памяти в ГМАРСП используются прямо адресуемые регистровые памяти с намного меньшим объемом, расположенные в тех блоках архитектуры ГМАРСП, где они востребованы:

– 4×16 бит для хранения констант и 16×52 бит для временного хранения операндов в памяти совпадений каждой вычислительной секции;
– 8×16 бит для хранения констант и 16×32 бит для подгружаемых в процессе обработки капсулы данных в вычислителе каждой секции.

Тем самым устраняется узкое место классических потоковых архитектур. Запись и чтение операндов в память на основе сравнения теговых полей выполняется в ГМАРСП в несколько раз быстрее.

2. Последовательные участки кода выполняются в УП без потери времени на сравнение тегированных маркеров и с более высокой скоростью, чем в рекуррентном обработчике сигналов (РОС): частота работы РОС на ПЛИС не превышает 25 МГц, в то время как частота синхронизации УП NIOS-II составляет 160 МГц, а ARM Cortex-A9 — 1,5 ГГц. Кроме того, в ряде случаев есть возможность исполнить последовательный код в УП одновременно с подготовкой к выполнению в РОС параллельного участка кода (загрузка конфигурационных параметров, констант и т. д.).
3. В каждый момент времени выполняется только одна итерация цикла. Это резко сокращает размер тегов, потери времени на их коммуникацию по сети и объем аппаратуры. Если в потоковых системах массового параллелизма разрядность теговых полей в несколько раз превышает полезную разрядность обрабатываемых данных, то в ГМАРСП число разрядов теговых полей (36) сравнимо с полезной разрядностью обрабатываемых данных: 16 и 38 в неупакованных операндах и 64 в упакованных.

Предложенные способы реализации хранилищ данных обеспечивают предпосылки для реанимации потоковой концепции в области ЦОС, стоимостный фактор которой делает нецелесообразным прямолинейное использование в ней технических решений из области потоковых систем массового параллелизма. Они не требуют сложных аппаратных механизмов для поддержки присущей ГМАРСП капсульной модели программирования.

3 Аппаратная модель ГМАРСП

Структура аппаратной модели ГМАРСП представлена на рис. 1. Она включает в себя УП, шину адреса/данных и РОС. К шине адреса/данных могут быть подключены дополнительные устройства памяти и интерфейса, относящиеся к управляющему уровню.

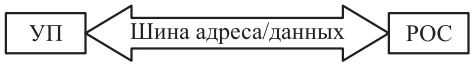

Рис. 1 Структура аппаратной модели ГМАРСП

В рамках методологии построения архитектуры ГМАРСП роль УП может играть традиционный (фон-неймановский) процессор любого типа. Главное

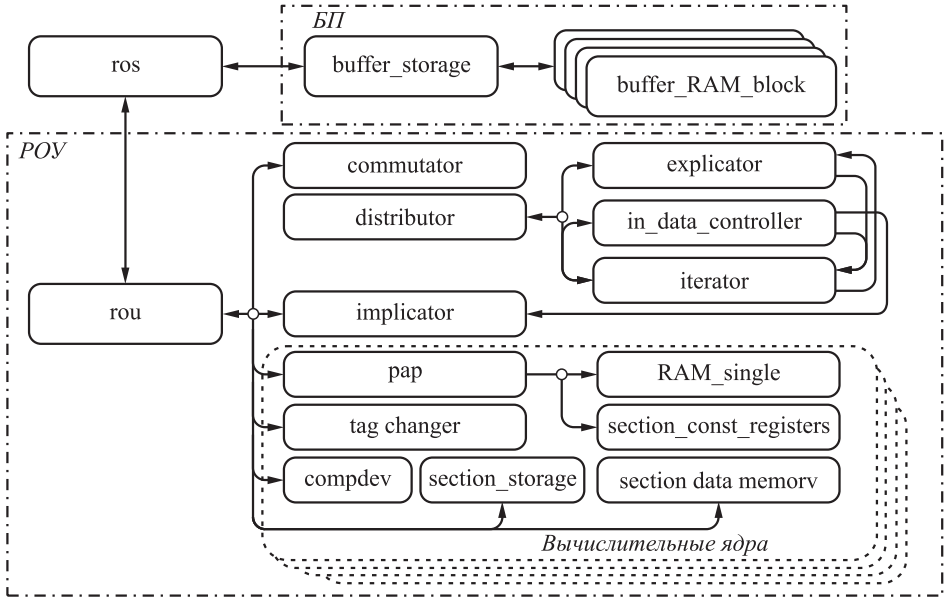

Рис. 2 Структура VHDL-модели РОС

требование к его характеристикам — достаточное быстродействие, обеспечивающее требуемую обработку результатов РОС и своевременную подкачку данных в РОС. В текущей реализации ГМАРСП на ПЛИС Intel Arria10 SoC функции УП выполняет программный процессор NIOS-II или аппаратный процессор ARM Cortex-A9. NIOS-II интегрируется в проект ГМАРСП в виде готового сложного функционального блока, имеющегося в библиотеке системы автоматизированного проектирования (САПР) Quartus Prime SE 18 (Intel) [3]. ARM Cortex-A9 изначально присутствует в ПЛИС на используемой отладочной плате HAN Pilot Platform.

Блок РОС реализован в виде модели уровня регистровых передач на языке VHDL. Целиком аппаратная модель, изображенная на рис. 1, собирается в приложении Qsys, запускаемом из оболочки САПР Quartus.

Структура аппаратной модели РОС на языке VHDL представлена на рис. 2. Она состоит из главного модуля **ros** и модулей, реализующих функционал РОС. Модуль верхнего уровня **ros** организует взаимодействие буферной памяти (БП, **buffer_storage**) и рекуррентного обрабатывающего устройства (РОУ, **rou**) и содержит арбитр обращений к БП со стороны УП (чтение и запись) и РОУ (запись). Запись в БП со стороны РОУ имеет высший приоритет.

Особенности организации работы блока БП:

(1) БП состоит из 4 банков памяти и контроллера БП, управляющего обменом данными между БП, РОУ и УП. Один операнд капсулы располагается

в четырех банках БП по одному базовому адресу. Каждый операнд сопровождается битом готовности, разрешающим чтение операнда из БП в РОУ;

(2) банки БП представляют собой двухпортовые ОЗУ. Первый порт служит для записи данных в БП со стороны УП и РОУ и для чтения данных из БП в УП. Он управляется системной частотой шины адреса/данных. Второй порт используется для чтения операндов из БП в РОУ и управляется внутренней частотой РОУ;

(3) 64-разрядные операнды записываются в БП из УП и считываются из БП в УП по шине адреса/данных за две операции записи, по 32 разряда за одну операцию;

(4) чтение из БП в РОУ и запись выходных и промежуточных результатов из РОУ в БП выполняется одним 64-битным операндом.

Аппаратная модель РОС реализована в виде кольцевого конвейера, ступени которого используют микроконвейеры для обработки данных в течение одного вычислительного шага (ВШ) длительностью четыре периода тактовой частоты РОУ.

Ступень 1 включает контроллер БП и массив ячеек памяти БП, заполняемый из УП капсулами реализуемых алгоритмов (модуль **buffer_storage**). Контроллер БП считывает по 4 операнда за один ВШ и передает их в РОУ попарно.

Ступень 2 включает контроллер входных данных (КВД, модуль **in_data_controller**). Здесь считанные из БП операнды распаковываются (при необходимости), рассортировываются и передаются в экспликатор (ступень 3) группой из 1–4 операндов. Операнды, предназначенные для итератора (ступень 3) и импликатора (ступень 6), передаются по назначению.

Ступень 3 включает экспликатор (модуль **explicator**) и итератор (модуль **iterator**). Экспликатор распределяет операнды по вычислительным секциям. Итератор организует циклическое использование операндов.

Ступень 4 включает секционные памяти совпадений (ПС, модуль **pap**): ПС подготавливает операнды для обработки в секционном вычислителе и имеет доступ в секционные памяти констант.

Ступень 5 включает секционный вычислитель (модуль **compdev**) и преобразователь тегов (ПТ, модуль **tag_changer**). Они вычисляют содержательную и функциональную части нового операнда.

Ступень 6 включает коммутатор (модуль **commutator**) и импликатор (модуль **implicator**). Коммутатор комплектует операнды результатов обработки капсулы и при необходимости отправляет их в экспликатор для дальнейшего использования внутри РОУ. Импликатор вылавливает в потоке результирующих операндов промежуточные и окончательные выходные данные и отправляет их в БП.

Память операндов есть не только в блоке БП, но и в блоках КВД и экспликаторе в виде FIFO (first in, first out). Она обеспечивает безостановочную

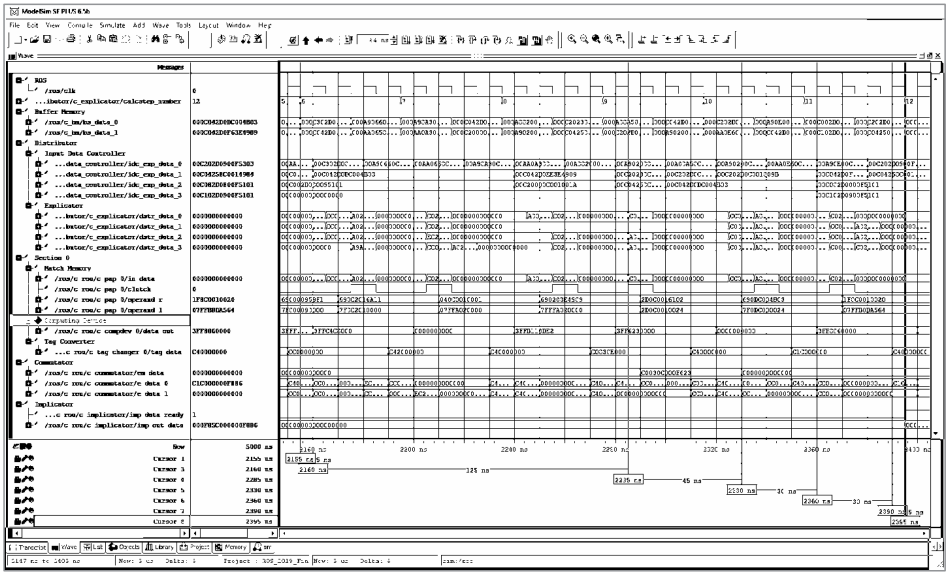

Рис. 3 Временная диаграмма работы ГМАРСП

работу конвейера РОУ во время распаковки упакованных операндов в блоке КВД и накопление операндов в экспликаторе для своевременного наполнения конвейера РОУ.

Дополнительные блоки памяти **section_const_registers**, **section_storage** и **section_data_memory** в каждой вычислительной секции хранят константы, записываемые УП и используемые при обработке капсул.

Аппаратная модель РОУ написана на языке VHDL с использованием подмножества синтезируемых операторов. Это обеспечивает ее прямую трансляцию из алгоритмического описания в аппаратный базис средствами логических синтезаторов. На ПЛИС фирмы Intel логический синтез выполняется в САПР Quartus.

Рисунок 3 иллюстрирует работу ГМАРСП. Вторая строка диаграммы показывает номера ВШ. Курсоры на временно́й оси отмечают движение операнда по конвейеру РОС от момента считывания из БП до появления результата его обработки на выходе импликатора. Третий слева курсор показывает время поступления операнда в ПС (MatchMemory), где он сцепляется (ассоциируется) с хранящимся там другим операндом (clutch = 1) и формирует пару операндов для секционного вычислителя (четвертый курсор). Результат вычисления (пятый курсор) попадает в коммутатор в качестве содержательной части нового операнда, который подается на E-шину (e_data_0, шестой курсор), отлавливается импликатором и передается в БП (седьмой курсор).

Таким образом, латентность конвейера в данном случае равна 6 ВШ или 24 тактам частоты синхронизации РОС (сигнал clk). Как видно из рис. 3, дольше всего операнд обрабатывался в экспликаторе из-за того, что в экспликаторе есть входное FIFO, в котором операнд как минимум два ВШ продвигался к выходу из экспликатора. Эту ситуацию можно охарактеризовать как предвыборку входных операндов из БП и не учитывать указанные два ВШ при определении латентности конвейера РОС. Рисунок 3 подтверждает, что ПС не является узким местом ГМАРСП.

4 Реализация ГМАРСП на ПЛИС

Аппаратные ресурсы, потребовавшиеся для реализации ГМАРСП в ПЛИС в режиме автоматического синтеза с помощью САПР Quartus [3], представлены в таблице для двух вариантов реализации УП: программного процессора NIOS-II и аппаратного процессора ARM Cortex-A9. Рисунок 4 демонстрирует результат синтеза варианта ГМАРСП с NIOS-II.

Сравнение аппаратных ресурсов для двух вариантов УП показывает следующее:

– ресурсы для аппаратной реализации РОС при использовании программного и аппаратного процессора управляющего уровня примерно одинаковы;

– дополнительные ресурсы (около 5% от общего объема использованных ресурсов ПЛИС в случае УП NIOS-II и около 9% в случае ARM Cortex-A9) требуются для аппаратной реализации интерфейса РОС с УП и периферии.

Аппаратные ресурсы при реализации ГМАРСП

Блоки ГМАРСП	Ресурсы ПЛИС с NIOS-II (N) и ARM Cortex-A9 (A)									
	ALMs		Registers		Memory bits		M20K Blocks		DSP Blocks	
	N	A	N	A	N	A	N	A	N	A
Буферная память	5 153	8 609	10 097	10 113	141 900	141 900	16	16	0	0
Контроллер входных данных	1 852	2 025	1 109	1 107	0	0	0	0	0	0
Экспликатор	13 565	12 539	2 687	2 698	0	0	0	0	0	0
Итератор	1 249	1 372	601	640	0	0	0	0	0	0
Память совпадений	4 317	4 192	3 888	3 703	3 200	3 200	12	12	0	0
Вычислитель	18 252	13 953	1 920	2 938	0	0	0	0	4	4
Коммутатор	626	442	589	380	0	0	0	0	0	0
Преобразователь тегов	45	37	92	94	0	0	0	0	0	0
Импликатор	1 416	1 174	653	653	0	0	0	0	0	0
Секционная память	14 212	14 298	20 888	20 888	2 048	2 048	8	8	0	0
Управляющий уровень	3 641	6 675	4 048	9 254	33 788 672	33 555 968	2 080	2 051	4	0
Итого:	**64 328**	**65 316**	**46 746**	**51 840**	**33 935 820**	**33 703 116**	**2 106**	**2 087**	**8**	**4**

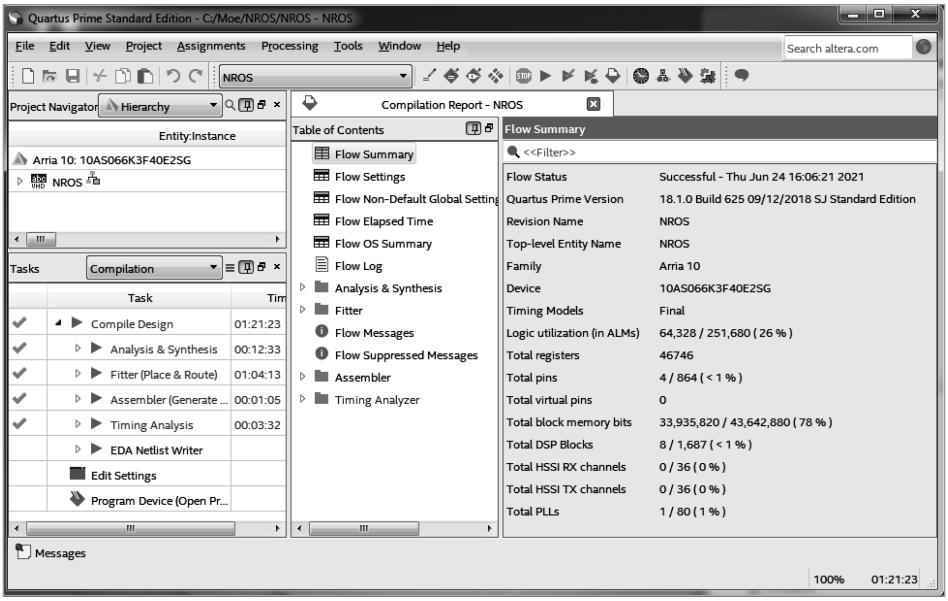

Рис. 4 Скриншот результатов синтеза ГМАРСП с УП NIOS-II

Реализация ГМАРСП на ПЛИС преследовала цель апробации рекуррентной архитектуры в аппаратном виде и получения сравнительных оценок функциональных возможностей рекуррентной архитектуры и ее производительности на синтетических тестах и реальных приложениях ЦОС. ПЛИС обеспечивает быстрое проектирование и корректировку любого проекта за счет программирования готовых аппаратных ресурсов — адаптивных логических блоков, параметризуемых блоков памяти, аппаратных умножителей и т. д., но не претендует на достижение быстродействия, обеспечиваемого в заказной технологии.

Относительно небольшой объем использованных ресурсов ПЛИС (26%) от общего объема ресурсов ПЛИС позволит в дальнейшем расширить функциональность ГМАРСП за счет увеличения числа секций или встраивания аппаратной реализации типовых алгоритмов, например быстрого преобразования Фурье.

5 Заключение

Предложенные и реализованные особенности архитектурной организации ГМАРСП повысили эффективность использования потоковой парадигмы в приложениях ЦОС, сократив требуемый объем аппаратурных затрат и повысив ее производительность. ПЛИС ограничивает производительность реализованных на ней приложений. Однако программные средства отладки и верификации при-

ложений на ПЛИС позволяют оценить производительность последних в числе операций на один такт частоты синхронизации.

Аппаратная модель рекуррентной части ГМАРСП на уровне регистровых передач на языке VHDL обеспечила верификацию архитектуры ГМАРСП на макетном образце на базе отладочной платы HAN Pilot Platform с ПЛИС Intel Arria10 SoC 10AS066K3F40E2SG.

Аппаратные ресурсы, требуемые для реализации текущей архитектуры ГМАРСП на ПЛИС, подтверждают возможность использования современных отладочных плат для макетирования следующих поколений ГМАРСП с более широкой функциональностью.

Литература

1. *Shikunov Yu., Stepchenkov Yu., Khilko D.* Recurrent mechanism developments in the data-flow computer architecture // IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering Proceedings. — Piscataway, NJ, USA: IEEE, 2018. P. 1413–1418. doi: 10.1109/EIConRus.2018.8317362.
2. HAN Pilot Platform. Specifications. https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=216&No=1133&PartNo=2.
3. Intel Quartus Prime Software. https://www.intel.ru/content/www/ru/ru/software/programmable/quartus-prime/download.html.

Поступила в редакцию 25.06.21

RECURRENT SIGNAL PROCESSOR HARDWARE IMPLEMENTATION

Yu. A. Stepchenkov, N. V. Morozov, Yu. G. Diachenko, and D. V. Khilko

Federal Research Center “Computer Science and Control” of the Russian Academy of Sciences, 44-2 Vavilov Str., Moscow 119133, Russian Federation

Abstract: The paper presents the results of hybrid architecture of recurrent multicore signal processor (HARMSP) hardware implementation as register transfer level VHDL-model and its prototype approbation on a development board with Intel Arria10 field-programmable gate array (FPGA). HARMSP consists of von-Neumann master processor at control architecture level and data-flow recurrent processor with four computing sections at operational level. Hardware HARMSP model is a complex of software or hardware control processor (CP) implementation and operational level VHDL-model. CAD Quartus (Intel) provides the software CP implementation on FPGA, whereas SoC FPGA on the development board contains the hardware CP implementation as dual-core Cortex-A9 processor.

Keywords: recurrent signal processor; multicore hybrid architecture; VHDL-model; FPGA

DOI: 10.14357/08696527210310

Acknowledgments

The research was supported by the Russian Science Foundation (project No. 19-11-0034).

References

1. Shikunov, Yu., Yu. Stepchenkov, and D. Khilko. 2018. Recurrent mechanism developments in the data-flow computer architecture. *IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering Proceedings*. Piscataway, NJ: IEEE. 1413–1418. doi: 10.1109/EIConRus.2018.8317362.
2. HAN Pilot Platform. Specifications. Available at: https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=216&No=1133&PartNo=2 (accessed August 31, 2021). см русский вариант, он другой
3. Intel Quartus Prime Software. Available at: https://www.intel.ru/content/www/ru/ru/software/programmable/quartus-prime/download.html (accessed August 31, 2021).

Received June 25, 2021

Contributors

Stepchenkov Yuri A. (b. 1951) — Candidate of Science (PhD) in technology, leading scientist, Institute of Informatics Problems, Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences, 44-2 Vavilov Str., Moscow 119333, Russian Federation; YStepchenkov@ipiran.ru

Morozov Nikolai V. (b. 1956) — senior scientist, Institute of Informatics Problems, Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences, 44-2 Vavilov Str., Moscow 119333, Russian Federation; NMorozov@ipiran.ru

Diachenko Yuri G. (b. 1958) — Candidate of Science (PhD) in technology, senior scientist, Institute of Informatics Problems, Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences, 44-2 Vavilov Str., Moscow 119333, Russian Federation; diaura@mail.ru

Khilko Dmitri V. (b. 1987) — senior scientist, Institute of Informatics Problems, Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences, 44-2 Vavilov Str., Moscow 119333, Russian Federation; dhilko@yandex.ru