УДК 004.31:004.032.34 DOI: 10.31114/2078-7707-2021-2-77-82

Аппаратная верификация рекуррентного обработчика сигналов на ПЛИС

Ю.А. Степченков, Н.В. Морозов, Ю.Г. Дьяченко, Д.В. Хилько,
Д.Ю. Степченков, Ю.И. Шикунов

Институт проблем информатики Федерального исследовательского центра "Информатика и управление" Российской академии наук (ФИЦ ИУ РАН)

{YStepchenkov, NMorozov, YDiachenko, DHilko, DStepchenkov, YShikunov}@ipiran.ru

***Аннотация* — В работе представлены результаты верификации аппаратной реализации гибридной многоядерной архитектуры рекуррентного сигнального процессора (ГМАРСП), представленной VHDL-моделью уровня регистровых передач. Макетный образец реализован на отладочной плате HAN Pilot Platform с программируемой логической интегральной схемой (ПЛИС) Intel Arria10 SoC 10AS066K3F40E2SG с помощью системы Quartus Pro 18 (Intel). ГМАРСП включает ведущий фон-неймановский процессор в качестве управляющего уровня и потоковый процессор с четырьмя вычислительными ядрами в качестве операционного уровня. В составе макетного образца управляющий процессор (УП) реализуется либо программно (NIOS II), либо аппаратно (ARM Cortex-A9). Тестирование аппаратной реализации ГМАРСП на типовом приложении цифровой обработки данных − распознавателе изолированных слов (РИС) – на отладочной плате подтвердило ее битэкзектность имитационной модели ГМАРСП и исходной C++ модели РИС. Достигнутая производительность аппаратной реализации ГМАРСП обеспечивает работу РИС на отладочной плате в режиме реального времени. Верификация аппаратной реализации ГМАРСП на синтетических тестах, покрывающих основную часть алгоритмов цифровой обработки сигналов, показала, что ее производительность в среднем на 5% превышает производительность процессора обработки цифровых данных C55x фирмы Texas Instruments.**

***Ключевые слова*— рекуррентный сигнальный процессор; гибридная многоядерная архитектура; VHDL-модель; ПЛИС; распознаватель изолированных слов.**

I. Введение

Гибридная многоядерная архитектура рекуррентного сигнального процессора (ГМАРСП) [1, 2] является альтернативой традиционной вычислительной архитектуре. Она относится к классу потоковых архитектур и реализует принцип объединения в одном операнде данных и кода операции, выполняемой над этими данными. Совокупность таких операндов образует капсулу, предназначенную для выполнения определенного алгоритма и аналогичную программному коду для традиционных вычислительных архитектур.

В процессе обработки капсулы в ГМАРСП операнды подвергаются рекуррентной развертке, характер которой определяется как исходным видом операндов, так и логикой работы рекуррентных преобразователей в составе аппаратных средств ГМАРСП. В связи с этим основным средством разработки и отладки капсул для ГМАРСП служит имитационная модель ГМАРСП [3, 4], учитывающая особенности реализации ГМАРСП. Она учитывает особенности архитектурной организации ГМАРСП и позволяет разработчику капсулы проследить за продвижением данных по этапам работы алгоритма, их распределением по параллельным вычислительным ресурсам и последовательностью рекуррентных преобразований.

Имитационная модель ГМАРСП облегчает процесс капсульного программирования: написания капсулы и ее отладки. Однако она учитывает не все технические особенности аппаратной реализации архитектуры ГМАРСП. Кроме того, ее нельзя напрямую использовать в качестве исходного алгоритмического описания ГМАРСП для ее логического и топологического синтеза 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 имитационной модели и позволяющей оценить характеристики ГМАРСП в режиме реального времени на отладочной плате, является 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 [5], ее технические характеристики и результаты тестирования, подтверждающие идентичность функционирования аппаратной модели на отладочной плате и имитационной модели ГМАРСП на компьютере с точки зрения результатов, получаемых на всех этапах обработки данных. В качестве субъекта тестирования были выбраны алгоритмы приложения Распознаватель изолированных слов из предметной

области, алгоритмы которой являются типовыми для любых задач цифровой обработки данных.

II. Аппаратная модель ГМАРСП

Структура аппаратной модели ГМАРСП представлена на рис. 1. Она включает в себя Управляющий Процессор (УП), шину адреса/данных и Рекуррентный Обработчик Сигналов (РОС), представленный Контроллером буферной памяти (КБП), Буферной памятью (БП), Распределителем, Интерфейсом обмена данными и N параллельными Вычислительными секциями (ВС). КБП содержит арбитр обращений к БП со стороны УП (чтение и запись) и РОУ (запись). Запись в БП со стороны РОУ имеет высший приоритет. К шине адреса/данных могут быть подключены дополнительные устройства памяти и интерфейса, относящиеся к управляющему уровню.

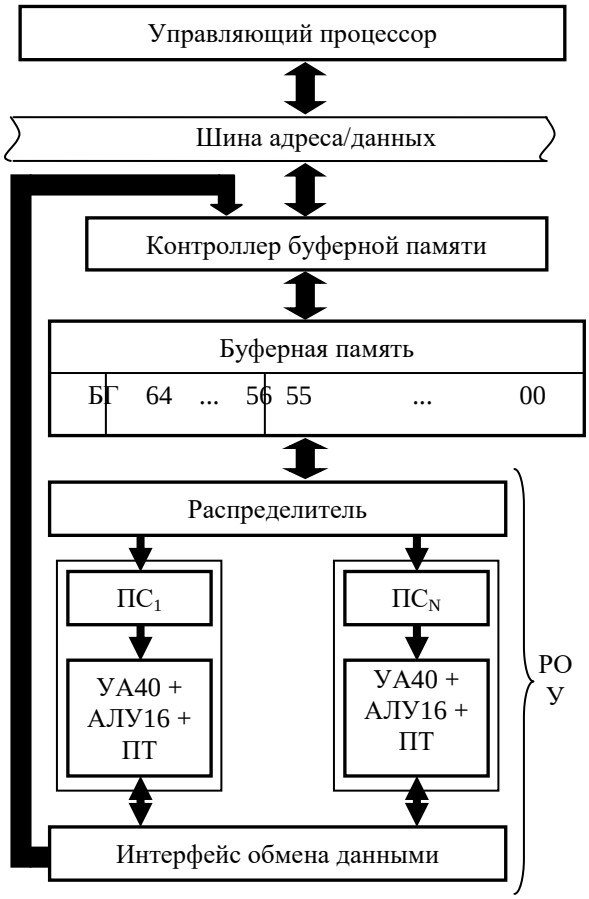

Рис. 1. Структура аппаратной модели ГМАРСП

Каждая ВС включает Память совпадений (ПС) и Вычислитель, имеющий в своем составе Умножитель с 40-разрядным аккумулятором (УА40) и аппаратным устройством сдвига вправо и влево на 1-15 разрядов, 16-разрядное арифметико-логическое устройство (АЛУ16) и Преобразователь тэгов (ПТ). Вместе с Распределителем и Интерфейсом обмена данными ВС составляют Рекуррентное обрабатывающее устройство (РОУ).

В рамках методологии построения архитектуры ГМАРСП роль УП может играть традиционный (фон-Неймановский) процессор любого типа. Главное требование к его характеристикам – достаточное быстродействие и функциональность, обеспечивающие требуемые темп и характер обработки промежуточных результатов РОС и записи данных в РОС. В частности, на УП возлагается выполнение операции деления, которая в ГМАРСП пока аппаратно не реализована. В текущей реализации ГМАРСП на ПЛИС Intel Arria10 SoC 10AS066K3F40E2SG функции УП выполняет программный процессор NIOS-II, интегрируемый в проект на этапе логического синтеза в САПР Quartus Prime SE 18 [6], или процессор ARM Cortex-A9, реализованный аппаратно в ПЛИС.

Особенности организации работы блока БП:

1) БП состоит из 4 банков памяти общей разрядности 65 бит. Один операнд капсулы располагается по частям в четырех банках БП по одному базовому адресу. Старший бит операнда хранит признак его упакованности. В отличие от остальных операндов, используемых в ГМАРСП, упакованный операнд не имеет функциональных полей. Он состоит из четырех 16-разрядных содержательных полей, назначение и принцип использования которых определяется предшествующими операндами капсулы. Отдельный регистр хранит биты готовности (БГ) всех операндов капсулы. Считывать операнд из БП в РОУ разрешается только в том случае, если его БГ установлен. БГ устанавливается либо УП при записи всего операнда капсулы или его содержательной части в БП, либо при записи результата обработки капсулы из РОУ.

2) Банки БП являются двухпортовыми 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.

Блок РОС реализован в виде аппаратной модели уровня регистровых передач на языке VHDL. Его функциональные модули организованы в виде кольцевого конвейера, изображенного на рис. 2. УП записывает исходную капсулу в БП и дает сигнал старта. После этого РОУ считывает капсулу из БП и обрабатывает ее данные в соответствии с инструкциями, содержащимися в функциональных полях операндов, до встречи с операндом-финишером.

Ступень 1 конвейера включает контроллер БП и массив ячеек памяти БП. Операнды капсул записываются в БП из УП. Контроллер БП считывает из БП по одному операнду за один такт и передает их во вторую ступень конвейера попарно.

Ступень 2 включает Контроллер Входных Данных (КВД). Он распаковывает считанные из БП операнды (при необходимости) и передает в следующую ступень конвейера (Экспликатор) группой из 1 – 4 операндов. Одновременно КВД выделяет в потоке операндов те, которые предназначены для Итератора (ступень 3) и Импликатора (ступень 6), и передает их по назначению.

Ступень 3 включает Экспликатор и Итератор. Экспликатор распределяет операнды по вычислительным секциям и передает их в следующую ступень конвейера. Итератор организует циклическое использование команд управления процессом обработки данных.

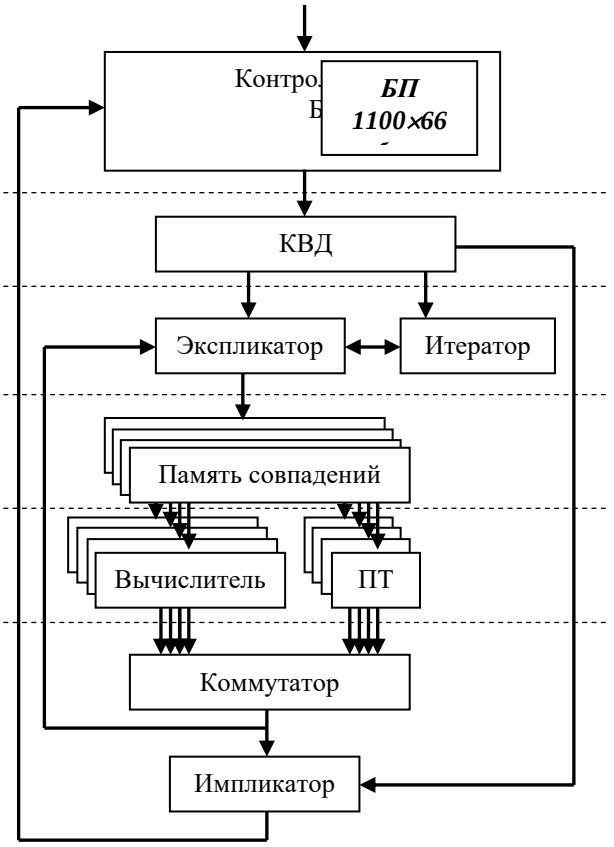

Рис. 2. Кольцевой конвейер РОС

Ступень 4 включает четыре секционные Памяти Совпадений (ПС), которые подготавливают пары операндов для выполнения простой или суперскалярной операции в Вычислителе своей секции. Источником одного операнда пары служит операнд из ПС, а второго операнда – Экспликатор.

Ступень 5 включает четыре секционных Вычислителя и четыре ПТ. Они вычисляют содержательную и функциональную части нового операнда соответственно.

Ступень 6 включает Коммутатор и Импликатор. Первый комплектует новые операнды из содержательной части, полученной в ВС, и функциональной части, сформированной в ПТ, и при необходимости отправляет их обратно в Экспликатор для использования в дальнейших вычислениях. Импликатор выделяет в потоке операндов, формируемых Коммутатором, промежуточные и окончательные результаты обработки капсулы и отправляет их в БП через КБП.

Память операндов есть не только в блоке БП, но и в блоках КВД и Экспликаторе в виде регистровых файлов с кольцевыми счетчиками адресов. Она обеспечивает безостановочную работу конвейера РОУ во время распаковки упакованных операндов в блоке КВД и накопление операндов в Экспликаторе для своевременного наполнения конвейера РОУ. При заполнении регистрового файла накачка операндов в него приостанавливается. Опустошение регистрового файла, в свою очередь, приостанавливает работу конвейера РОУ.

III. Реализация ГМАРСП на ПЛИС

Реализации ГМАРСП на ПЛИС преследовала цель апробации рекуррентной архитектуры в аппаратном базисе. ПЛИС, как известно, обеспечивает быстрое проектирование, корректировку и отладку любого проекта за счет программирования готовых аппаратных ресурсов – настройки их на выполнение требуемых функций. Такая реализация принципиально не может обладать высоким быстродействием, т.к. характеризуется большой аппаратной избыточностью, но очень удобна для выявления "узких мест" в разрабатываемой цифровой схеме. Реальное быстродействие проекта на ПЛИС определяется ее тактовой частотой. Но производительность архитектуры, заложенной в аппаратную модель, можно оценить в тактах частоты РОУ, необходимых для обработки капсулы алгоритма.

Аппаратные ресурсы, потребовавшиеся для реализации ГМАРСП в ПЛИС Intel Arria10 SoC 10AS066K3F40E2SG в режиме автоматического синтеза с помощью САПР Quartus Prime SE 18.0 [6] представлены в табл. 1 и 2 для двух вариантов реализации УП: программного процессора NIOS-II (табл. 1) и аппаратного процессора ARM Cortex-A9 (табл. 2). Аппаратные затраты определены в количестве адаптивных логических модулей (АЛМ), регистров, битов памяти и аппаратно реализованных блоков DSP. Сравнение аппаратных ресурсов для двух вариантов УП показывает, что ресурсы для реализации РОС примерно одинаковы в обоих случаях. Это естественно, так как РОС реализуется единообразно в обоих случаях с использованием аппаратных ресурсов ПЛИС, и только УП требует разных ресурсов для своей реализации.

Таблица 1

Аппаратные ресурсы при реализации ГМАРСП с NIOS-II

Функциональные блоки ГМАРСП	**Ресурсы ПЛИС**			
	АЛМ	***Регистры***	***Память, бит***	***DSP***
Буферная Память	15201	10049	141900	0
Контроллер Входных Данных	1852	1109	0	0
Экспликатор	13565	2687	0	0
Итератор	1249	601	0	0
Память Совпадений	4317	3888	3200	0
Вычислитель	18252	1920	0	4
Коммутатор	626	589	0	0
Преобразователь Тегов	45	92	0	0
Импликатор	1416	653	0	0
Секционная память	14212	20888	2048	0
Управляющий уровень	3641	4048	33788672	4
Итого:	74440	46698	33935820	8

Таблица 2

Аппаратные ресурсы при реализации ГМАРСП с ARM Cortex-A9

Функциональные блоки ГМАРСП	**Ресурсы ПЛИС**			
	АЛМ	***Регистры***	***Память, бит***	***DSP***
Буферная Память	12791	9587	141900	0
Контроллер Входных Данных	2025	1107	0	0
Экспликатор	12539	2698	0	0
Итератор	1372	640	0	0
Память Совпадений	4192	3703	3200	0
Вычислитель	13953	2938	0	4
Коммутатор	442	380	0	0
Преобразователь Тегов	37	94	0	0
Импликатор	1174	653	0	0
Секционная память	14298	20888	2048	0
Управляющий уровень	6675	9254	33555968	0
Итого:	75356	51314	33703116	4

Относительно небольшой объем использованных ресурсов ПЛИС позволяет в дальнейшем расширить функциональность ГМАРСП за счет увеличения числа секций и/или встраивания аппаратной реализации специальных алгоритмов, например, быстрого преобразования Фурье.

IV. Тестирование аппаратной модели

Для верификации аппаратной модели ГМАРСП на отладочной плате было выбрано типовое приложение цифровой обработки сигналов – Распознаватель изолированных слов. Целью верификации было доказать идентичность результатов, получаемых имитационной моделью ГМАРСП [3] и аппаратной моделью на ПЛИС.

Прогон капсул, сгенерированных для всех алгоритмов Распознавателя по исходным данным, извлеченным тестовых из произнесений 100 слов из словаря Распознавателя, в имитационной и аппаратной моделях подтвердил бит-экзектность всех промежуточных результатов. Суммарная точность распознавания 96%, показанная аппаратной моделью на отладочной плате, совпала с точностью распознавания, обеспечиваемой исходной C++ программой Распознавателя на персональном компьютере.

В табл. 3 приведены оценки производительности аппаратной модели ГМАРСП в сравнении с DSP C55x при выполнении синтетических тестовых алгоритмов [7], являющихся типовыми для широкого класса задач цифровой обработки сигналов и аналогичных тестам из набора [8]. При этом быстродействие DSP C55x принято за эталонное, а производительность ГМАРСП приведена по отношению к DSP C55x. Здесь: K_{FIR} – число коэффициентов фильтра; K_{BQ} – число секций фильтра (биквадов).

Таблица 3

Производительность ГМАРСП в сравнении с DSP C55x на синтетических тестах

Алгоритм	***DSP C55x***	***ГМАРСП***
Фильтр с конечной импульсной характеристикой для реальных данных	1	$1+\frac{1}{1+K_{FIR}}$
Фильтр с конечной импульсной характеристикой для одиночных отсчетов	1	1
Фильтр с конечной импульсной характеристикой для комплексных данных	1	$1+\frac{3}{1+K_{FIR}}$
Адаптивный фильтр наименьших среднеквадратичных значений	1	$1-\frac{K_{FIR}-1}{4+3\cdot K_{FIR}}$
Би-квадратный фильтр с бесконечной импульсной характеристикой	1	$1+\frac{3+10\cdot K_{QB}}{4+21\cdot K_{QB}}$
Сумма поразрядных произведений двух векторов	1	1
Поразрядная сумма двух векторов	1	1,50
Поиск максимума в векторе	1	0,60
Декодер Витерби	1	0,31
256-точечное быстрое преобразование Фурье (БПФ)	1	1,25

Анализ табл. 3 показывает, что ГМАРСП существенно лучше, чем DSP C55x, выполняет алгоритмы фильтрации (за исключением адаптивного фильтра) и некоторые другие. Например, при $K_{FIR} = 6$ и $K_{BQ} = 3$ ГМАРСП работает в 1,14 – 1,50 раза быстрее, чем DSP C55x. Однако на некоторых задачах ГМАРСП работает в 1,67 – 3,23 раза медленнее, чем DSP C55x. Это связано с наличием в последнем аппаратных средств поддержки соответствующих алгоритмов, существенно ускоряющих их выполнение аналогично тому, как аппаратно реализованный параллельный умножитель работает быстрее его программной реализации.

Отмеченный недостаток ГМАРСП может быть устранен за счет расширения ее функциональности:

- более широкой аппаратной поддержки суперскалярных операций в вычислительных секциях,

- применения особенностей технологии сверхдлинных команд (very long instruction word, VLIW), в том числе увеличения числа рабочих регистров в секционном Вычислителе,

- внедрения механизмов быстрой загрузки данных в регистры компьютерных блоков, минуя некоторые этапы конвейерной обработки,

Таким образом, экспериментальное тестирование и оценка производительности макетного образца ГМАРСП на отладочной плате HAN Pilot Platform, полученного с помощью его автоматического синтеза из исходного VHDL-описания аппаратной модели РОС и программирования в ПЛИС средствами САПР Quartus Prime SE 18.0, подтвердило эффективность архитектурных решений ГМАРСП. В среднем на синтетических тестах табл. 3 он продемонстрировал на 5% лучшую производительность, чем DSP C55x фирмы Texas Instruments.

Проверка работы приложения Распознаватель изолированных слов на базе ГМАРСП, аппаратно реализованной в ПЛИС, показала идентичность результатов, получаемых исходной C++ моделью распознавателя, имитационной моделью ГМАРСП и аппаратной моделью ГМАРСП на отладочной плате на всех промежуточных этапах вычислений. При соответствующей частоте синхронизации проект Распознавателя изолированных слов на ПЛИС обеспечивает работу приложения в режиме реального времени с сохранением необходимого качества распознавания.

V. Заключение

Аппаратная модель ГМАРСП на уровне регистровых передач на языке VHDL обеспечила проверку корректности архитектурных решений, заложенных в ГМАРСП, на макетном образце на базе отладочной платы HAN Pilot Platform с ПЛИС Intel Arria10 SoC 10AS066K3F40E2SG.

Тестирование аппаратной реализации ГМАРСП на типовом приложении цифровой обработки данных – распознавателе изолированных слов – на отладочной плате подтвердило ее битэкзектность имитационной модели ГМАРСП и исходной C++ модели РИС.

Сравнение временных затрат на выполнение алгоритмов цифровой обработки информации на макетном образце ГМАРСП и в DSP C55x Texas Instruments в тактах синхронизации подтвердило эффективность архитектурных решений ГМАРСП при выполнении задач, допускающих распараллеливание вычислений. В среднем производительность ГМАРСП в количестве тактов синхронизации, требующихся для выполнения вычислений, оказалась на 5% выше производительности DSP C55x.

Тестирование аппаратной модели на отладочной плате HAN Pilot Platform позволило определить "узкие места" архитектуры ГМАРСП, "расшивка" которых позволит дополнительно увеличить ее производительность. Это будет предметом нашей дальнейшей работы.

Поддержка

Исследование выполнено при финансовой поддержке гранта Российского научного фонда (проект 19-11-00334) в Институте проблем информатики ФИЦ ИУ РАН.

Литература

[1] Yu. Shikunov, Yu. Stepchenkov, D. Khilko. Recurrent mechanism developments in the data-flow computer architecture // 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus) Moscow, Russia, 29 Jan.-1 Feb., 2018. —IEEE, P. 1413 – 1418. DOI: 10.1109/EIConRus.2018.8317362.

[2] Степченков Ю.А., Дьяченко Ю.Г., Хилько Д.В., Петрухин В.С. Рекуррентная потоковая архитектура: особенности и проблемы реализации // Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС). 2016. № 2. С. 120-127.

[3] Хилько Д.В., Степченков Ю.А., Шикунов Ю.И., Орлов Г.А. Развитие средств капсульного программирования потоковой рекуррентной архитектуры // Проблемы разработки перспективных микро- и наноэлектронных систем.(МЭС). 2018. Вып. 3. С. 2-9. doi:10.31114/2078-7707-2018-3-2-9.

[4] Д.В. Хилько, Ю.А. Степченков. Теоретические аспекты разработки методологии программирования рекуррентной архитектуры // Системы и средства информатики, – М.: ТОРУС ПРЕСС, Т. 23, № 2, 2013 – С. 133-153.

[5] HAN Pilot Platform. Specifications. URL: https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=216&No=1133&PartNo=2. (дата последнего обращения 20.05.2021г.)

[6] Intel Quartus Prime Software. URL: https://www.intel.ru/content/www/ru/ru/software/programmable/quartus-prime/download.html (дата последнего обращения 20.05.2021г.)

[7] Yury A. Stepchenkov, Dmitry V. Khilko, Yury I. Shikunov, Georgy A. Orlov. DSP Filter Kernels Preliminary Benchmarking for Recurrent Data-flow Architecture // 2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus) St. Petersburg, Moscow, Russia, January 26-29, 2021. P. 2040-2044.

[8] The BDTImark2000™: A Measure of DSP Execution Speed. Berkeley Design Technology, Inc. URL: http://meseec.ce.rit.edu/eecc722-fall2001/papers/dsp/4/bdtimark2000.pdf (дата последнего обращения 24.05.2021г.).

Hardware verification of the recurrent signal processor on FPGA

Y.A. Stepchenkov, N.V. Morozov, Y.G. Diachenko, D.V. Khilko,
D.Y. Stepchenkov, Y.I. Shikunov

Institute of Informatics Problems, Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences (IPI FRC CSC RAS), IPI RAS

{YStepchenkov, NMorozov, YDiachenko, DHilko, DStepchenkov, YShikunov}@ipiran.ru

***Abstract* — Paper represents Hybrid Architecture of Recurrent Multi-core Signal Processor (HARMSP) hardware implementation results. It describes HARMSP's register transfer level model in VHDL and hardware prototype on HAN Pilot Platform demo-board with field-programmable gate array (FPGA) Intel Arria10 SoC 10AS066K3F40E2SG. HARMSP consists of a von Neumann master processor on a control level and a dataflow processor on an operational level. Dataflow processor includes four computing cores. HARMSP's hardware model combines program or hardware implementation of the controlling processor (CP) and VHDL model of the operational level. CP's program implementation is a default option provided by Quartus software (Intel) for FPGA. FPGA Intel Arria10 SoC on demo-board provides CP's hardware implementation as Cortex-A9 two-core processor. Testing the HARMSP's hardware prototype on demo-board using an isolated word recognizer as a typical data processing application has proved that the hardware model is bit-exact with HARMSP's imitation model. The HARMSP's hardware prototype's achieved performance ensures isolated word recognizer's operation in real-time mode on demo-board. It is slightly better than the performance of the C55x (Texas Instruments) digital signal processor.**

***Keywords* — recurrent signal processor; hybrid multi-core architecture; VHDL model; FPGA; isolated word recognizer**

REFERENCES

[1] Yu. Shikunov, Yu. Stepchenkov, D. Khilko. Recurrent mechanism developments in the data-flow computer architecture // 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus) Moscow, Russia, 29 Jan.-1 Feb., 2018. IEEE, P. 1413-1418. DOI: 10.1109/EIConRus.2018. 8317362.

[2] Stepchenkov Yu.A., Diachenko Yu.G., Khilko D.V., Petrukhin V.S. Recurrent data-flow architecture: features and realization problems // Problems of Perspective Micro- and Nanoelectronic Systems Development - 2016. Proceedings / edited by A. Stempkovsky, Moscow, IPPM RAS, 2016. Part 2. P. 120-127. (in Russian).

[3] Khilko D.V., Stepchenkov Yu.A., Shikunov Yu.I., Orlov G.A. Development of Capsule Programming Means for Recurrent Data-flow Architecture // Problems of Perspective Micro- and Nanoelectronic Systems Development. 2018. Issue 3. P. 2-9. doi:10.31114/2078-7707-2018-3-2-9

[4] D.V. Hilko_ Yu.A. Stepchenkov. Teoreticheskie aspekti razrabotki metodologii programmirovaniya rekurrentnoi arhitekturi (Theoretical aspects of developing a recurrent architecture programming methodology) // Sistemy i sredstva informatiki – M., TORUS PRESS, V. 23, No 2, 2013 – P. 133-153 (in Russian).

[5] HAN Pilot Platform. Specifications. URL: https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=216&No=1133&PartNo=2. (last access date 20.05.2021)

[6] Intel Quartus Prime Software. URL: https://www.intel.ru/content/www/ru/ru/software/programmable/quartus-prime/download.html (last access date 20.05.2021)

[7] Yury A. Stepchenkov, Dmitry V. Khilko, Yury I. Shikunov, Georgy A. Orlov. DSP Filter Kernels Preliminary Benchmarking for Recurrent Data-flow Architecture // 2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus) St. Petersburg, Moscow, Russia, January 26-29, 2021. P. 2040-2044.

[8] The BDTImark2000™: A Measure of DSP Execution Speed. Berkeley Design Technology, Inc. URL: http://meseec.ce.rit.edu/eecc722-fall2001/papers/dsp/4/bdtimark2000.pdf (last access date 24.05.2021).