УДК 004.383.3; 004.272.44

КОНФИГУРИРУЕМЫЙ ВЫЧИСЛИТЕЛЬ НА БАЗЕ FPGA ДЛЯ ПОТОКОВОЙ ОБРАБОТКИ ВИДЕОСИГНАЛОВ

***Сизов М.М.**, инженер-программист Института автоматики и электрометрии СО РАН, e-mail: sizov.m.m@gmail.com;*
***Зюбин В.Е.**, д.т.н., ведущий научный сотрудник Института автоматики и электрометрии СО РАН, e-mail: zyubin@iae.nsk.su.*

CONFIGURABLE FPGA ARCHITECTURE FOR VIDEO STREAM PROCESSING

Sizov M.M., Zyubin V.E.

In this paper we present an analysis of current FPGA development approaches, from that analysis requirements for a video stream processing algorithms development system are deduced. We proposed an approach for implementing of such system based on hybrid FPGA + ARM SoC and a configurable pipelined FPGA architecture for video stream processing. System's implementation consists of predefined algorithm blocks, and a Tcl-based IDE extension which transforms high-level configuration file to a complete VHDL project and automates building and flashing process. The system was designed with the goal of simplifying a development of high-performance stream processing applications, but in the same time system reduces set of problems which can be easily solve. This drawback could be mitigated as the system is extensible and uses standard AXI4-Stream protocol.

Key words: FPGA, analysis requirements, video stream, VHDL, ARM SoC.

Ключевые слова: гибридные архитектуры, конвейерная обработка видеосигнала, ПЛИС, FPGA, VHDL.

Анализируются существующие решения по разработке вычислителей на базе FPGA для потоковой обработки видеосигналов с учетом специфики рассматриваемой области, формулируются требования к системе создания алгоритмов потоковой обработки видео, предлагается архитектура конфигурируемого конвейерного вычислителя, ориентированного на обработку видеосигнала, и описывается реализация предложенного подхода, ориентированного на аппаратную платформу Zynq.

Введение

Цифровые системы обработки видеосигналов – одно из важнейших направлений в информационных технологиях, связанных с автоматизацией технологических процессов, бесконтактными измерениями, контролем качества и др. Источник сигнала в таких системах – цифровая камера, формирующая поток видеокадров в сотни мегабайт в секунду. Для обработки таких потоков данных используются подходы, предполагающие параллелизм и конвейеризацию [1]. Особый интерес у исследователей вызывают алгоритмы на основе конвейера, поскольку позволяют начать обработку информации, не дожидаясь конца кадра, «на лету». Для реализации таких алгоритмов используются программируемые вентильные матрицы (Field Programmable Gate Array, или FPGA) [2], которые в дополнение привлекательны своей компактностью, низким энергопотреблением и возможностью контроля периферийного оборудования.

При всех достоинствах разработка систем на базе FPGA трудоемка. При программировании используются специализированные языки, так называемые языки описания аппаратуры (VHDL, Verilog), программы пишутся в терминах триггеров, регистров и их соединений; а существующие ограничения на объем памяти, находящийся на FPGA, и сложность реализации пользовательского интерфейса, как правило, приводят к необходимости взаимодействия с процессором общего назначения.

Заметно снижает трудоемкость разработки использование гибридных архитектур, совмещающих на одном кристалле FPGA и ARM-процессор. В существующих на данный момент гибридных архитектурах FPGA связаны с процессором через шину данных AMBA AXI [3]. Ведущие производители FPGA компании Xilinx и Altera предлагают семейства гибридных кристаллов Zynq [4] и Arria V [5], соответственно. Эти кристаллы используют сторонние производители. Так плата Zedboard (Digilent) на основе кристалла Zynq имеет стандартный для ПК набор периферии и FMC-коннектор (FPGA mezzanine card) с возможностью подключения высокоскоростной видеокамеры [6]. Программирование поддержано специализиро-ванным IDE, включающем средства трансляции VHDL/Verilog, генерации файла-прошивки, загрузки, эмуляции. Платформа обеспечивает возможность проведения как исследовательских работ, так и создание практических приложений в области цифровых систем обработки видеосигналов.

Хотя производители облегчают реализацию взаимодействия с внешним ОЗУ, предоставляя программные драйверы и IP-ядра с реализацией механизма DMA [7], сложность программирования взаимодействия процессора и FPGA остается высокой. Это стимулирует исследователей искать новые подходы, и в первую очередь предметно-ориентированные, снижающие трудоемкость программирования FPGA.

В статье анализируются существующие решения для разработки на FPGA с учетом специфики рассматриваемой области, формулируются требования к системе

создания алгоритмов потоковой обработки видео, предлагается архитектура конфигурируемого конвейерного вычислителя, ориентированного на обработку видеосигнала, и описывается реализация предложенного подхода, ориентированного на аппаратную платформу Zynq.

Высокоуровневые подходы к разработке алгоритмов на FPGA

LabVIEW и Matlab. Общая тенденция развития инструментов для программирования FPGA – увеличение уровня абстракции понятий, которыми оперирует программист. В качестве примеров можно привести программные пакеты LabVIEW [8] и Matlab [9], которые в дополнение к средствам программирования процессоров общего назначения имеют расширения, предназначенные для программирования FPGA штатными средствами (языки G, Simulink, Matlab). Недостатки использования этих средств – отсутствие возможностей профилирования и анализа генерируемого VHDL-кода, необходимость приобретать кроме специализированных расширений еще и базовые пакеты, содержащие с точки зрения программирования FPGA заведомо избыточную функциональность, за которую приходится платить. Что касается LabVIEW, то это решение еще и ориентировано на оборудование National Instruments. Matlab позволяет генерировать код на языках аппаратуры из программ на языках Simulink (конечные автоматы и диаграммы потока данных) и Matlab (матрицы). Matlab позволяет использовать оборудование основных производителей FPGA (Altera, Xilinx), в принципе допускает работу с Zynq, но не поддерживает потоковый интерфейс передачи данных (AXI-Stream), что снижает общую пропускную способность создаваемых систем.

Языки управления потоком исполнения. Существующие решения Handel-C [10], Impulse-C [11], Vivado High-Level Synthesis [12] используют Си-подобный синтаксис для создания алгоритма и последующей трансляции в языки описания аппаратуры. К недостаткам подхода относят разницу в парадигмах: язык Си относится к языкам управления потоком исполнения, поэтому эффективное использование возможностей FPGA предполагает наличие у программиста знаний архитектуры FPGA и преобразований, выполняемых транслятором на этапе кодогенерации.

Dataflow подход. Dataflow-парадигма более привлекательна для программирования FPGA. Программа описывается сетью взаимодействующих функциональных модулей (Actor) через однонаправленные каналы связи, данные представлены токенами. По каналам связи передаются также управляющие токены, например, токен начала кадра изображения или токен начала строки. В классическом варианте модули ожидают данных на всех входных каналах связи, проводят вычисления и после этого передают результаты на выход модуля. Однако для увеличения производительности системы используется конвейерная обработка данных.

Примером использования Dataflow-парадигмы с конвейерной архитектурой является язык Caph [13], он позволяет описывать функциональные модули и описывать их связи, затем транслирует внутреннее представление в HDL-код.

FPGA обладают тремя уровнями памяти (триггеры 10-100 Кб, SRAM-блоки 100-1000 Кб, внешняя DDR-память, к которой имеется доступ либо через пины общего назначения, либо через механизмы DMA), при написании программы на языке Caph нет явных механизмов указания типа используемой памяти, что приводит к случаям неэффективного распределения памяти и сужению класса решаемых задач при обработке видео, также не решается проблема взаимодействия с процессором общего назначения. Caph использует функциональную парадигму программирования, что также увеличивает требования к квалификации программиста.

В результате проведенного анализа были сформулированы требования к системе разработки алгоритмов потоковой обработки видеосигналов на FPGA:

1. Использование системы не должно требовать специализированных знаний об устройстве FPGA и языков описания аппаратуры.

2. Должна быть предусмотрена возможность описания взаимодействия с процессором общего назначения.

3. Результирующий код должен иметь формат, допускающий визуальный анализ, конкретно VHDL.

Конфигурируемый вычислитель

Для решения проблем взаимодействия с процессором общего назначения предлагается использовать гибридные архитектуры.

Алгоритм обработки потокового видео предлагается описывать в виде конфигурируемого вычислителя – последовательно соединенных библиотечных блоков (алгоритмических примитивов), образующих тракт обработки данных (рис. 1).

Рис. 1: Схема тракта обработки данных

Соединение блоков однонаправленно и осуществляется по стандартному протоколу AXI-Stream [14], ориентированному на потоковую обработку сигналов. При этом однозначно определены ядро-передатчик данных (master) и ядро-приемник (slave).

Начальный источник данных для тракта – IP-ядро (от Intellectual Property) видеокамеры, расположенный в FPGA, или процессор общего назначения (использующий DMA). По завершению IP-ядро DMA сохраняет результат обработки в ОЗУ процессора.

В качестве алгоритмических примитивов выступают: фильтр свертки (размытие, резкость), гамма-корректор, медианный фильтр, конечный автомат.

Конечный автомат задается наборами состояний и переменных, а также правилами перехода между состояниями. Состояние представляет собой действия, выполняемые при получении новой порции данных: арифметические и логические операции с переменными, проверка значений переменных, конца строки и конца кадра.

Переменные конечного автомата доступны со стороны процессора общего назначения по протоколу AXI-Lite и могут хранить как дополнительную информацию, так и флаги, например, флаг необходимости сохранения кадра на жестком диске.

Реализация конфигурируемого вычислителя

Предложенный подход был реализован для платы Zedboard с гибридным кристаллом Zynq 7000 [4] и CMOS камеры VITA-2000. Плата имеет 512 Мб ОЗУ, 10/100/1000 Ethernet, VGA, HDMI, USB-JTAG, USB-UART, Audio I/O, PMOD-интерфейс, слот SD-карты. Взаимодействие платы и видеокамеры организовано через интерфейс FMC (FPGA Mezzanine Card).

FPGA часть кристалла Zynq обеспечивает взаимодействие с периферией и позволяет реализовывать пользовательские алгоритмы, в том числе, ресурсоемкие вычисления. ARM-процессор выполняет функции процессора общего назначения. С платой поставляется среда разработки Vivado (Xilinx) и драйверы для двух операционных систем – Linux и XilKernel. Среда разработки Vivado обеспечивает возможность создания пользовательских программ на языках VHDL, Verilog, Си, средства отладки и симуляции. Ядро системы Vivado написано на языке Tcl и допускает консольное управление проектом на этом языке.

Система создания алгоритмов обработки потокового видео включает программу «генерации проекта и загрузки в целевую плату» (Tcl), шаблон интерфейсной программы для процессора общего назначения (Си) и библиотеку ал-горитмических примити-вов (VHDL). Библиотека алгоритми-ческих примитивов подключе-на к системе Vivado как пользовательский репозиторий. Алгоритмические примитивы реализуют протокол передачи видеопотока поверх стандартного интерфейса AXI (AXI4-Stream Video IP [15]), каждый примитив взаимно-однозначно связан с одноименной Tcl-функцией. Алгоритмические примитивы разрабатываются, моделируются и отлаживаются с помощью стандартных инструментов, входящих в поставку IDE Vivado.

Создаваемые алгоритмические примитивы должны поддерживать протокол AXI4-Stream Video IP [15] и заданный формат видеопотока (разрешение кадра, количество байтов на пиксель). Список доступных алгоритмических примитивов указывается в Tcl-скрипте.

Создание алгоритма обработки потокового видео ведется на отдельном ПК и состоит из следующих шагов:

1. В текстовом редакторе задается вычислительный тракт – последовательность алгоритмических примитивов в виде списка параметризованных Tcl-функций и описания конечного автомата на языке VHDL с использованием AXI-Stream шаблона.

2. Запускается программа «генерации проекта и загрузки в целевую плату», которая через консольное управление IDE Vivado:

– создает проект и блок-схему (Block Design) для FPGA, задает используемые IP-ядра DMA и видеокамеры;

– подключает IP-ядра алгоритмических примитивов в блок-схему проекта;

– запускает компилятор с автоматической трассировкой и прошивает полученный bitstream-файл в FPGA;

– создает код для XilKernel по шаблону, после чего производит его компиляцию и загрузку в ОЗУ ARM-процессора.

Заключение

В работе был предложен подход и реализована система создания алгоритмов обработки потокового видео в виде конфигурируемого конвейерного вычислителя.

Конвейерный вычислитель описывается как последовательность соединенных алгоритмических примитивов. Созданное на языке Tcl программное расширение IDE Vivado автоматически преобразует описание вычислителя в исполняемый код для платформы Zedboard (гибридный кристалл Zynq компании Xilinx).

К ограничениям предлагаемого метода можно отнести ориентированность на архитектуру Zynq, существование задач, использующих функции, не входящие в библиотеку алгоритмических примитивов, и автоматическую трассировку IDE Vivado, которая, может привести к рассинхронизации сигналов и невозможности размещения проекта на заданном кристалле. В этих случаях хотя и наблюдается сокращение времени выполнения проекта, возникает необходимость либо в расширении библиотеки, либо в ручной трассировке, и, как следствие, необходимость использовать высококвалифицированных программистов. В случае же, когда ограничения не проявляются, предлагаемый подход снижает и квалификационные требования к программисту.

Подход также может быть применим в случае гибридных кристаллов компании Altera, предоставляющей IDE Quartus с консольным Tcl-интерфейсом.

В будущем планируется расширить библиотеку алгоритмических примитивов, добавить в систему средства, упрощающие генерацию конечных автоматов, и ввести возможность графического описания вычислителя.

Литература

1. Аминев Д.А., Фокин А.Н. Методы селекции кадрового синхроимпульса для ввода несжатого видеопотока от однонаправленной одноразрядной цифровой линии и их реализация на ПЛИС // Цифровая обработка сигналов. 2014. №1. С. 52-55.

2. Панкратов В.Г., Карих А.А., Панфилов В.Н., Гуров А.Д. Вычисление функции неопределенности для пассивной локации на ПЛИС и графическом процессоре // Цифровая обработка сигналов. 2014. №1. С. 56-65.

3. AXI AMBA [Электронный ресурс] // Ссылка: http://www.arm.com/products/system-ip/amba-specifica-tions.php

4. Xilinx Zynq [Электронный ресурс] // Ссылка: http://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html

5. Altera Arria V [Электронный ресурс]//Ссылка: http://www.altera. com /devices/fpga/arria-fpgas/arria-v/hard-processor-system/arrv-sochps.html.

6. Crockett L. H. et al. The Zynq Book: Embedded Proces-sing with the Arm Cortex-A9 on the Xilinx Zynq-7000 All Programmable Soc. – 2014.

7. Лысаков К.Ф., Шадрин М.Ю. Особенности применения аппаратных устройств на базе FPGA для задач потоковой обработки изображений //Вестник Новосибирского государственного университета. Серия: Информационные технологии. – 2009. – Т. 7. – №. 3. – С. 15-22.

8. FPGA [Электронный ресурс] // Ссылка: http://www.ni.com/labview/fpga.

9. Matlab FPGA design [Электронный ресурс] // Ссылка: http://www.mathworks.com/solutions/fpga-design/

10. Loo S.M. et al. Handel-C for rapid prototyping of VLSI coprocessors for real time systems //System Theory, 2002. Proceedings of the Thirty-Fourth Southeastern Symposium on. – IEEE, 2002. – С. 6-10.

11. Antola A. et al. A novel hardware/software codesign methodology based on dynamic reconfiguration with Impulse C and CoDeveloper //Programmable Logic, 2007. SPL'07. 2007 3rd Southern Conference on. – IEEE, 2007. – С. 221-224.

12. Vivado High Level Synthesis [Электронный ресурс] // Ссылка: http://www.xilinx.com/products/design-tools /viva-do/integration/esl-design.html

13. Serot J., Berry F., Ahmed S. Implementing stream-processing applications on FPGAs: a DSL-based approach //Field Programmable Logic and Applications (FPL), 2011 International Conference on. – IEEE, 2011. – С. 130-137.

14. AMBA AXI4-Stream Protocol Specification v1.0 [Электронный ресурс] // Ссылка: http://infocenter.arm.com/help /index.jsp?topic=/com.arm.doc.ihi0051a/index.html.

15. AXI4-Stream Video IP and System Design Guide [Электронный ресурс] // Ссылка: http://www.xilinx.com/sup-port/documentation/ip_documentation/axi_videoip/v1_0/ug934_axi_videoIP.pdf.