ПОСТРОЕНИЕ СИНТЕЗИРУЕМОГО ПРОЦЕССОРНОГО ЯДРА С ПЕРЕОПРЕДЕЛЯЕМЫМ НАБОРОМ КОМАНД НА БАЗЕ FPGA

Матюшин П.М.

ФГУП Воронежский НИИ Связи,
394018, г. Воронеж, ул. Плехановская д.14, e-mail: matushin@imail.ru

Рассмотрены варианты построения SoC систем на основе FPGA. Предложен метод построения процессорного ядра с переопределяемым набором инструкций. Подробно описан вариант реализации процессорного ядра с предложенной архитектурой на FPGA Xilinx.

1. Введение

Последнее время во всем мире значительно возрос интерес к построению “Систем На Одном Кристалле” (SoC – System-On-a-Chip). Основой большинства таких систем являются процессорные ядра, некоторый объем программируемой логики и периферийные устройства. Все основные производители микросхем программируемой логики предлагают свои решения для построения процессорных ядер и SoC систем. Компания Xilinx предлагает как 32-битные RISC ядра MicroBlaze [1], так и простые 8-битные контроллеры PikoBlaze [2]. В самой передовой серии FPGA Virtex II Pro добавлены уже аппаратные ядра IBM PowerPC [3]. Компания Altera разработала систему Excalibur [4] с поддержкой программного ядра Nios и аппаратного ядра ARM. В микросхемах серии FPSLIC компании Atmel [5] объединены аппаратное ядро микроконтроллера AVR и FPGA емкостью до 40 тысяч вентилей. Кроме собственных разработок компаний, выпускающих программируемую логику, сейчас на рынке представлены процессорные ядра других фирм [6-8].

В большинстве своем, предлагаемые ядра имеют жесткую архитектуру и ограниченные настройки. Возможности программируемой логики не используются в них в полном объеме.

В этой работе предлагается способ построения синтезируемых процессорных ядер на базе FPGA, позволяющий получать настраиваемые под конкретные задачи системы, дает больше возможностей использования программируемой логики и существенно экономит ресурсы, затраченные на разработку.

2. Новый подход к построению архитектуры процессорных ядер

Главной задачей при разработке ядра было получение достаточно мощного управляющего устройства при малом объеме, занимаемом в FPGA. Кроме того, устройство должно было выполнять ряд операций, реализация которых в виде последовательности команд привела бы к значительному увеличению объема программы и снижению скорости работы ядра. В то же время, реализация этих операций в виде отдельных логических блоков не требовала больших ресурсов FPGA, уменьшала объем программы и не снижала скорости работы ядра. Но в этом случае нужен был сложный механизм управления этими блоками.

Выходом из этой ситуации стало разделение набора инструкций на три группы: внутренние, внешние и адресные инструкции. Адресные инструкции (команды условного и безусловного перехода, вызова и выхода из подпрограмм) выполняются контроллером указателя адреса памяти программ. Внутренние инструкции (логические операции, команды переноса, сдвиг, запись константы и др.) выполняются АЛУ. Внутренние инструкции изменяют состояния регистров общего назначения (РОН), с которыми работают. Время выполнения одинаково для всех внутренних инструкций.

Внешние инструкции выполняются отдельными исполняющими модулями. Ядро только подает данные на входы и считывает результаты с выходов исполняющих модулей. Входы и выходы модулей являются регистрами ядра. Модули являются “черными ящиками”, их внутренняя структура не важна для ядра. Это могут быть модули для выполнения как простых операций: сложение, вычитание, вычисление модуля, ограничение значения числа в заданном диапазоне, так и более сложных: нормализация, умножение, вычисление математических функций. Время выполнения внешних инструкций может быть различно для разных инструкций.

Такой механизм позволяет создать процессорное ядро с переопределяемым набором команд. Для каждой новой задачи мы можем реализовать новые операции, при этом синтаксис ассемблера и интерфейс ядра и исполняющих модулей остается без изменений. Существенно упрощается проектирование новых приложений и сокращается время разработки.

3.Реализация 16-битного процессорного ядра на FPGA Xilinx

Рассмотрим процессорное ядро, построенное на предложенных принципах, описанное на языке VHDL для FPGA Virtex компании Xilinx. За основу была выбрана 16-разрядная архитектура с отдельной памятью для данных и программ (гарвардская архитектура). На Рис.1 представлена блок-схема ядра.

Память программ в данной реализации имеет объем 4 Кбайта и реализована на структурах блочной памяти FPGA Xilinx Virtex (8 блоков). Адрес в памяти программ задается контроллером указателя адреса. В режиме нормальной работы программы указатель адреса инкрементирует свое значение каждый цикл. При выполнении адресных инструкций в указатель записывается новое значение адреса.

Ядро имеет восемь 16-разрядных регистров общего назначения (РОН), организованных в файл регистров, и набор регистров специального назначения (регистры внешних исполняющих модулей, арбитра параллельного интерфейса, контроллера прерываний). Дешифратор команд определяет тип команды и подает содержимое нужных РОН на входы АЛУ или исполняющих модулей. АЛУ состоит из трех функциональных модулей: логический, арифметический и сдвиговый. Каждый отвечает за свою группу внутренних команд.

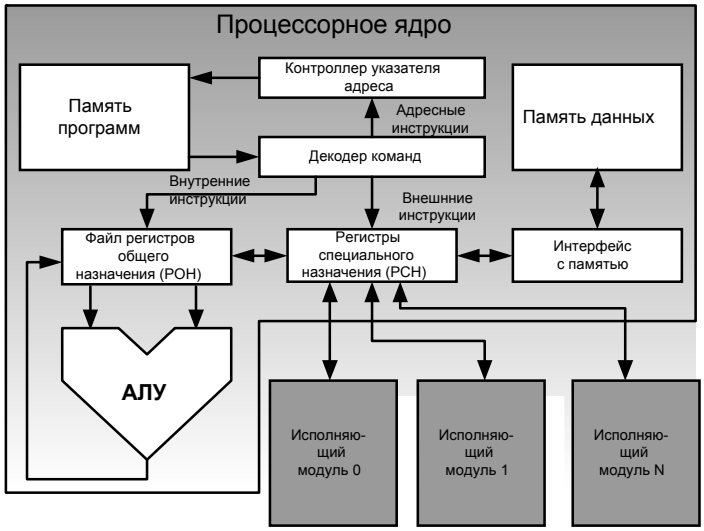

Рис.1 Архитектура процессорного ядра с переопределяемым набором инструкций

Для внешних исполняющих модулей определен интерфейс. Каждый модуль имеет два входа (UDn_I0, UDn_I1) и один выход (UDn_O). Здесь n – номер внешнего исполняющего модуля. Код внешней инструкции на ассемблере будет выглядеть так:

UDn POH(i), POH(j)

Результат выполнения внешней инструкции доступен по команде:

mov UDn_O, POH(i) - здесь **UDn_O** – источник (выход внешнего исполняющего модуля) и **POH(i)** – приемник.

Для обмена с памятью данных организован синхронный параллельный интерфейс. Размер адресуемой памяти до 8 Кбайт (в текущей реализации 2 Кбайта).

Ядро также содержит набор периферийных устройств: контроллер прерываний (поддерживает до 6-ти прерываний), программируемый таймер, загрузчик и отладчик программ. Для увеличения функциональных возможностей может быть добавлены другие специализированные периферийные устройства.

Хранение программы ядра может быть осуществлено как непосредственно в FPGA, так и во внешней памяти. При хранении программы в FPGA необходимо трассировать весь кристалл FPGA при любом изменении программы. Для сокращения времени, затрачиваемого на внесение изменений в программу, используется внешняя память (ПЗУ). В этом случае перед началом работы программа будет считываться из ПЗУ и загружаться в память программ ядра.

Ядро описано на языке VHDL, что позволяет, с небольшими изменениями кода, реализовать его на FPGA других фирм.

4. Заключение

В последние годы во всем мире бурно развивается концепция построения цифровых систем на одном программируемом кристалле. В большинстве таких систем используются процессорные ядра с традиционной архитектурой и небольшими возможностями настройки под конкретные приложения. Предложенный в данной работе вариант построения процессорного ядра позволяет в значительно большей степени использовать возможности программируемой логики и удешевить разработку цифровых электронных систем. Переопределяемый под каждую задачу набор инструкций, возможности изменения объема памяти и числа прерываний, добавление различных периферийных устройств дают возможность почти полной перестройки структуры без изменений печатных плат.

Процессорное ядро, построенное на принципах, предложенных в данной работе, успешно реализовано в приемопередатчике сложного широкополосного сигнала, разработанного Воронежским НИИ Связи.

Литература

1. Xilinx MicroBlaze Soft Processor http://www.xilinx.com/xlnx/xil_prodcat_product.jsp?title=microblaze
2. Xilinx PikoBlaze Soft Processor http://www.xilinx.com/ipcenter/processor_central/picoblaze/index.htm
3. Xilinx Virtex-II Pro™ Platform FPGA Handbook http://www.xilinx.com/publications/products/v2pro/handbook/index.htm
4. Altera Excalibur http://www.altera.com/literature/br/br_excalibur.pdf
5. Atmel FPSLIC™ http://www.atmel.com/atmel/products/prod39.htm
6. Atlas™ - Universal Digital Signal Computer http://www.eonic.com/
7. Digital Core Design http://www.dcd.com.pl/
8. ARM™ Processor Cores Overview http://www.arm.com/armtech/CPUs?OpenDocument

BUILDING FPGA BASED PROCESSOR SOFT CORE WITH REDEFINED INSTRUCTION SET

Matyushin P.

Voronezh Research Institute of Telecommunications (VRIT)
394018, 14 Plekhanovskaya st., Voronezh, Russia, e-mail: matushin@imail.ru

Abstract. Choices of FPGA based System-On-a-Chip (SoC) are described. A new method for building processor soft core with redefined instruction set is given.

The interest to FPGA based System-On-a-Chip (SoC) design is growing all over the world. Generally the processor soft cores used for SoC are characterized by a rigid architecture and limited abilities for user tunings [1-6]. The benefits of programming logic are not used in full scale. The present paper covers the design technique of the processor soft cores by FPGA, to obtain the systems tunes for particular purposes.

The main architecture design principle is the separation of a set of external user-defined instructions from all instructions. The external instructions are performed by the execution blocks, which are "black boxes" in respect to the processor soft core. The inputs and outputs of the execution blocks are the core registers. The execution blocks are able to perform simple operations, such as: absolute value calculation, number value limiting in the pre-set range, and more sophisticated operations, such as: normalization, multiplication, calculation of mathematic functions. Figure 1 shows the processor soft core architecture with the execution blocks.

Separation of the external instructions allows the creation of the processor soft core with a redefined set of commands. One can implement new operations by changing the execution blocks only for each new purpose. Moreover, the assembler syntax, the processor soft core interface, and the execution blocks remain unchanged. The design of new applications is simplified, and the time of design is reduced.

The design technique of the processor soft cores by FPGA with redefined number of instructions, changeable memory size and the number of interrupts, proposed in the present paper, provide the option of almost complete structural modification without changes of PCBs.

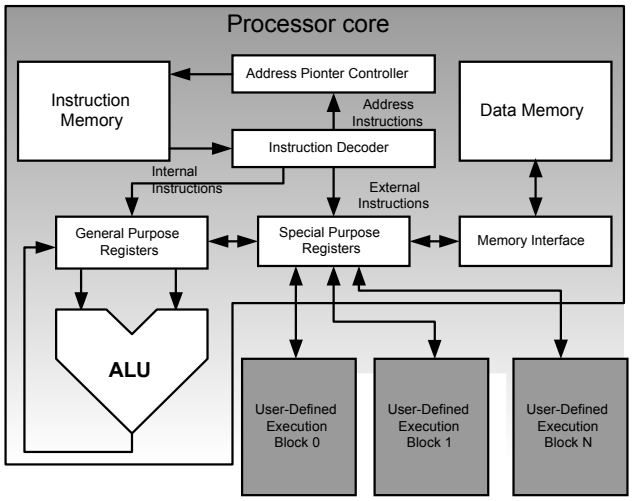

Figure 1. Processor soft core with redefined instruction set

References

1. Xilinx MicroBlaze Soft Processor http://www.xilinx.com/xlnx/xil_prodcat_product.jsp?title=microblaze
2. Xilinx Virtex-II Pro™ Platform FPGA Handbook http://www.xilinx.com/publications/products/v2pro/handbook/index.htm
3. Altera Excalibur http://www.altera.com/literature/br/br_excalibur.pdf
4. Atmel FPSLIC™ http://www.atmel.com/atmel/products/prod39.htm
5. Atlas™ - Universal Digital Signal Computer http://www.eonic.com/
6. Digital Core Design http://www.dcd.com.pl/