УДК 681.3.001.57

ПРОЕКТИРОВАНИЕ МИКРОПРОЦЕССОРА С РАСШИРЕННЫМ НАБОРОМ КОМАНД МАНИПУЛЯЦИИ БИТАМИ ДАННЫХ НА БАЗЕ АРХИТЕКТУРЫ OpenRisc1200

С. И. Назаров, А. В. Ляшенко, Л. С. Сотов*, А. Л. Хвалин*

ОАО «Институт критических технологий»
Россия, 410040, Саратов, пр. 50 лет Октября, 110А
E-mail: tantal@renet.ru

*Саратовский государственный университет
Россия, 410012, Саратов, Астраханская, 83
E-mail: kof@info.sgu.ru

В работе представлены разработка и реализация на базе архитектуры OpenRisc1200 микропроцессора OPBM с расширенным набором команд манипуляции битами данных. OPBM – 32-битный RISC микропроцессор, основанный на наборе команд ORBIS32. Производительность микропроцессора увеличивается от 2 до 10 раз благодаря возможности выполнять специальные инструкции побитовой обработки. Первые прототипы были разработаны на базе ПЛИС Xilinx XC6LX45. Показано, что по сравнению с существующими микропроцессорами у разработанного есть преимущества, заключающиеся в расширенной функциональности, улучшенной работе и пониженной сложности аппаратурных средств.

Ключевые слова: RISC-микропроцессор, инструкция перестановок, инструкции группирования и размещения битов, многоуровневая коммутационная сеть, процессор OpenRisc1200.

The Design and Implementation of the Advanced Bit Manipulation Microprocessor ORBM Based on the OpenRisc1200 Architecture

S. I. Nazarov, A. V. Lyashenko, L. S. Sotov, A. L. Khvalin

The design and implementation of the advanced bit manipulation microprocessor ORBM based on the OpenRisc1200 architecture is presented. ORBM is a 32 bit processor based on the ORBIS32 instruction set. The microprocessor performance is increased from two to ten times because microprocessor can execute special instructions of bit manipulation. The first prototypes were designed on the Xilinx XC6LX45 FPGA. It is shown that in comparison with existing microprocessors the developed microprocessor has advantages including expanded functionality, the improved performance and the lowered hardware complexity.

Key words: RISC-microprocessor, permutation instruction, subword extract instruction, subword deposit instruction, multistage interconnection network, OpenRisc1200 architecture.

С расширением области применения средств вычислительной техники все чаще возникают задачи, связанные с формированием перестановок или манипуляций битами данных машинного слова [1]. Битовые перестановки сложны для программной реализации. В связи с этим в последние годы проводятся интенсивные исследования в области разработки устройств, ускоряющих обработку битов данных.

В работе [2] был предложен способ структурного синтеза устройств разбиения строки входных данных для реализации инструкций манипуляции битами данных. Устройства манипуляции битами данных просты при последовательной обработке машинного слова [3], однако при разработке устройств, выполняющих манипуляции с битами машинного слова параллельно, проблемой является достаточно высокая аппаратурная сложность блока декодирования битов управления переключателями [4, 5]. Аналогичная проблема возникает и при других подходах к разработке устройства, основанных, например, на использовании многоуровневой коммутационной сети *butterfly* [6]. В работе [7] показано, что при осуществлении логических и циклических сдвигов битов данных на базе обратной топологии сети *butterfly* удается существенно упростить декодер битов управления и увеличить его быстродействие. При этом для построения универсального модуля манипуляции битами требуется две сети *butterfly* и *ibutterfly*, реализующие прямые и обратные преобразования. В работах [8–11] предпринят ряд шагов по упрощению и повышению быстродействия устройств, осуществляющих манипуляции битами данных, а в работе [12] описана практическая реализация универсального устройства манипуляции битами данных.

В данной статье описаны результаты исследований по созданию процессорной системы на основе открытых аппаратурных решений и разработанного авторами уникального модуля для ускоренной манипуляции битами данных. Для разработки и моделирования использовалась программа Xilinx ISE [13]. Применяемый в микропроцессоре универсальный модуль ускоренной манипуляции битами данных описан в работах [12, 14–16]. Особенности функционирования некоторых узлов модуля подробно обсуждены в [17–19].

Для тестирования устройства перестановки битов по его описанию, приведенному в [14–16], была составлена имитационная модель на языках Verilog и SystemC [20, 21].

Наименование программного модуля перестановки битов данных – BM_32. Входом модуля является 32-разрядная шина X, на которую подаются данные. Выходом модуля является 32-разрядная шина, поименованная как Y.

Перестановка разрядов в представлении X осуществляется в соответствии с набором однобитовых указателей, описание которых приведено в работах [2, 12, 22, 23]. Конкретику перестановок определяют также входные данные, подаваемые:

– на 16-битовые шины (управление матрицей перестановок) C1, C2, C3, C4, C5;

– на 5-разрядную шину (задание числа сдвигов) A;

– на 32-разрядную шину (задание маскирования разрядов) F.

Для тестирования функций логического и циклического сдвигов на языке Verilog HDL была написана и отлажена программа, включенная в

проект с наименованием bm_32. Данный проект синтезирует электронную схему, на вход которой последовательно методом циклического перебора подаются 32-разрядные данные и конкретные управляющие сигналы.

В проекте bm_32 реализован алгоритм идентификации разрядов, обеспечивающий простоту и наглядность результатов работы сдвигового устройства BM_32. В соответствии с этим алгоритмом на вход модуля перестановок подаются последовательно 5 чисел, расположенных в таблице друг под другом, в двоичном и шестнадцатиричном форматах. Это чтение иллюстрируется в таблице.

Таблица

Тестовые данные для проверки команд логического сдвига в модуле BM_32

Номер двоичного разряда	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**
Число 1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
Число 2	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
Число 3	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1
Число 4	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1
Число 5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0

Номер двоичного разряда	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**Шестнадцатиричный формат чисел**
Число 1	0	1	0	1	0	1	0	1	0	1	0	1	0	**AAAAAAAA**
Число 2	0	1	1	0	0	1	1	0	0	1	1	0	0	**CCCCCCCC**
Число 3	1	0	0	0	0	1	1	1	1	0	0	0	0	**F0F0F0F0**
Число 4	1	1	1	1	1	0	0	0	0	0	0	0	0	**FF00FF00**
Число 5	0	0	0	0	0	0	0	0	0	0	0	0	0	**FFFF0000**

Нумерация разрядов в таблице указана в верхней строке в десятичном исчислении, а образуемые двоичные разряды этой нумерации указаны в 5 строках под ней. Числа, которые образуют эти 5 строк в шестнадцатиричной системе исчисления, указаны в правом столбце таблицы. Анализируя таблицу можно заключить, что столбцы образуют числа длиной 5 бит, которые можно интерпретировать как порядковый номер двоичного разряда в таблице.

Числа подобраны так, что, применяя одну и ту же команду сдвига ко всем 5 числам и выводя данные в таблице, можно проверить правильность выполнения операции сдвига.

В силу универсальности модуль перестановок BM_32 обеспечивает все реализуемые в обычных процессорах виды сдвигов, а также произвольные перестановки.

Выбор процессорного ядра

Для реализации ОРБМ необходимо было сделать выбор процессорного ядра. При этом особенно полезными оказываются системы с открытыми

исходными текстами, размещенные в Интернете. Они позволяют использовать уже созданные и проверенные решения в области проектирования RISC-микропроцессора.

В качестве базового решения был выбран процессор OpenRisc [24], так как в интернет-сайтах представлена вся необходимая информация для разработки. Эту информацию представляет компания OpenCores, на интернет-сайтах которой [25] можно найти все нужные программы или ссылки на них и исходные тексты последних моделей процессорных систем с открытой архитектурой, получивших название OpenRisc 1000.

Перефразируя намеченную выше цель, можно сказать, что конкретной областью наших интересов было проведение необходимых изменений в существующих текстах процессора открытой архитектуры OR1200 и создание дополнений к ним в виде новых необходимых элементов, ускоряющих и расширяющих проведение всех возможных перестановок битов в машинном слове.

Для определенности остановимся на 32-разрядном исполнении машинных слов процессора. Процессор OR1200 имеет гарвардскую архитектуру, что означает выделение разных блоков памяти для хранения инструкций и хранения данных. Все узлы процессора, включая внутреннюю память, написаны на языке Verilog HDL.

В качестве исходного проекта процессора OR1200 был взят проект ORSoC, представленный на сайте [25]. Для программы симулятора ISim фирмы Xilinx из всего проекта ORSoC, который представляет собой обширный набор файлов не только на языке Verilog HDL, но и на языке C, требуются только папки из раздела orpsocv2\rtl\verilog\or1200.

Из HDL (Hardware Description Language) описания процессора OR1200, которому соответствует файл верхнего уровня or1200_top, следует, что он состоит из 13 блоков, определенным образом соединенных между собой через порты входа и выхода, и содержащих внутри себя цепи обработки сигналов. Каждый блок представляет собой модуль, в котором в иерархическом порядке могут содержаться новые модули. Все входы/выходы модулей, а также сигналы, цепи их приема/передачи, обработки и преобразования и необходимые константы поименованы. С точки зрения файла верхнего уровня они имеют унифицированный характер. Особые константы, имеющие назначение параметров настройки, а также определяющие ту или иную условную трансляцию, в проекте вынесены в отдельный файл с именем or1200_defines.v, который содержится в разделе orpsocv2\rtl\verilog\include. Изменяя константы из этого файла, пользователь может настраивать выполняемые процессором команды, указывать особые свойства, изменяющие конфигурацию, упрощающие и ускоряющие его работу.

Для обеспечения обработки новых команд манипуляции битами данных изменения, в первую очередь, вносятся в модуль центрального процессорного устройства (ЦПУ). В папке RTL этому устройству соответствует файл or1200_cpu.

Как файл верхнего уровня, так и связанные с ним файлы, обеспечивающие его поведенческое описание процессора, расположены в отмеченной выше папке RTL. Модуль ЦПУ ядра OR1200, описываемый в тексте файла верхнего уровня, структурирован и объединяет в себе 13 функциональных подмодулей, работающих одновременно. С точки зрения внутреннего протокола связи внутри этих модулей и связи между ними унифицированы и представляют собой выработку и анализ определенных сигналов в соответствующих цепях, что необходимо учитывать. Для правильной обработки внутренних сигналов процессора не следует нарушать принятые в нем правила сопряжения. Данное обстоятельство составляет основную сложность внесения изменений в тексты.

Ввиду важности унифицированных элементов ЦПУ перечислим файлы, содержащие их точные описания на языке Verilog HDL:

– or1200_genpc (Generate PC);

– or1200_if (модуль выбора инструкций (Instruction fetch));

– or1200_ctrl (блок внутренних преобразований исполняемых инструкций);

– or1200_rf (блок регистров общего назначения внутри ЦПУ);

– or1200_operandmuxes (модуль мультиплексирования двух регистровых операндов чтения файла (Mux for two register file read operands));

– or1200_alu (арифметико-логическое устройство (АЛУ));

– or1200_mult_mac (модуль умножения верхнего уровня (Top level multiplier and MAC 32×32=64+64));

– or1200_sprs (модуль декодирования SPR адреса и доступа к регистрам SPR (Decoding of SPR addresses and access to SPRs));

– or1200_lsu (модуль, реализующий интерфейс обмена данными CPU и DC (Load/Store unit - Interface between CPU and DC));

– or1200_wbmux (модуль записи выходных данных конвеера CPU (CPU's write-back stage of the pipeline));

– or1200_freeze (модуль генерации сигналов фиксаций и остановок внутри процессора (Generates all freezes and stalls inside RISC));

– or1200_except (модуль обработки исключений внутри CPU (Handles all OR1K exceptions inside CPU block));

– or1200_cfgr (VR, UPR и регистры конфигурации (VR, UPR and Configuration Registers)).

Исполнительным звеном команд процессора является арифметико-логическое устройство (АЛУ). Остальные звенья обеспечивают формирование правильных, точных по назначению, времени и месту всех необходимых сигналов команд, их операндов и последующую обработку результата.

Команды сдвига в процессоре OR1200

Процессор OR1200 имеет унифицированную кодировку системы команд. Код размещается побитно в одном машинном слове. В ходе выборки

и декодирования команды все ее составляющие находят свое место на соответствующих цепях, значения сигналов на которых должны быть достоверны при наступлении фронта каждого тактового импульса posedge clk. Логически звенья отработки команд процессора «читают» в двоичном формате эти значения, в соответствии с которыми в АЛУ осуществляется выбор того или иного действия обработки.

Однако изначально система распознавания команд настроена на совершенно иной формат, причем для отдельной команды может быть предусмотрена своя дополнительная идентификация разрядов.

Так, для обозначения команд, осуществляющих операции сдвига «shift» в OR1200, предусмотрены поля, образуемые следующими «штатными» разрядами:

– [31:26] – общее поле кода команды, которое должно содержать значения ‘h2E или ‘h38, в зависимости от способа указания величины сдвига;

– [25:21] – указатель на регистр rD, содержащий результат;

– [20:16] – указатель на регистр rA , содержащий операнд;

– [15:11] – указатель на регистр rB, который не используется, если код команды ‘h2E;

– [10:8] –не используемые три двоичных разряда;

– [7:6] – поле дополнительного кода, уточняющего команду, длиной 3 двоичных разряда;

– [5:0] – константа сдвига либо число ‘h8, если код команды ‘h38.

В поле дополнительных кодов можно указать одну из следующих 4 команд сдвига:

– Shift Left Logical (логический сдвиг влево);

– Shift Right Logical (логический сдвиг вправо);

– Shift Right Arithmetic (арифметический сдвиг вправо);

– Rotate Right (циклический сдвиг вправо).

Модуль «ctrl» ЦПУ считывает поля инструкций [31:26] в регистр «alu_op», а поля [9:6] в регистр «alu_op2». Далее модуль ctrl транслирует биты указанных регистров на внутренние цепи ЦПУ с одноименными названиями. Сам модуль АЛУ внутренних регистров не имеет и все свои действия по исполнению инструкций осуществляет в соответствии с сигналами входных цепей.

Коды команд сдвига назначаются соответствующими записями пользователя в файле определений or1200_defines.v.

Всем командам сдвига на линиях «alu_op» соответствует общее целевое значение 5'b0_1000, а конкретным операциям сдвига на линиях «alu_op2» соответствуют значения 4'd 0, 4'd1, 4'd2, 4'd3.

Модуль operandmuxes, расположенный в ЦПУ, по соответствующим указателям в исходном коде команды процессора формирует, используя другие общедоступные ресурсы ЦПУ, истинные значения аргументов команды, выставляя их на линиях operand_a и operand_b. К этим общедоступным теперь линиям подключены внутренние линии a и b, расположенные в АЛУ.

Для отработки вопросов работы новых инородных модулей в составе ЦПУ процессора OR1200, а также для будущего сравнения особенностей работы прежнего процессора и модифицированного нами была составлена тестовая программа, реализующая тот же алгоритм идентификации разрядов после исполнения штатных команд сдвига, который применялся ранее для тестировании логических сдвигов данных машинного слова в модуле BM_32 в проекте bm_32. Отличия в реализации нового проекта от проекта bm_32 состоят в том, что все необходимые линии и формируемые сигналы взяты из уже реализованного ЦПУ процессора OR1200. Новый проект имеет название «TestALU», и ему соответствует отдельный директорий с тем же именем.

Текст демонстрационной программы «TestALU» не приводится, чтобы не перегружать статью. Эту программу легко составить, используя текст демонстрационной программы работы процессора OR1200 в новой конфигурации, со встроенным в него устройством перестановок.

В блоке АЛУ операции сдвига запрограммированы посредством «штатных» операторов сдвига языка Verilog HDL (a >> b и a << b), использующих свои библиотечные схемы. На этапе исследований изменить логику работы этих команд внутри принципиальной схемы процессора нам не представляется целесообразным. Для предлагаемых пользователю инструментов работы с процессором библиотечные схемы отработки сдвигов Verilog HDL являются «черным ящиком».

Для изменения работы процессора с его прежней схемой манипуляциями сдвига можно перепрограммировать в новые, «нештатные» операции сдвига либо исключить их целиком, подключив новый узел внутри ЦПУ. Но в любом случае, поскольку все узлы процессора работают параллельно, необходимо добавить новый, 14-й, по счету модуль в ЦПУ, и при обращении к нему лучше использовать новые команды. К тому же на одном и том же вновь собранном процессоре OR1200 лучше иметь возможность проверить действия разных команд сдвига.

Новое содержание команды пользователя «cust5»

Для своей работы новый модуль должен будет использовать необходимые и уже «приготовленные» процессором линии с циркулирующими в них общими «штатными» сигналами. Для кодировки нового набора команд сдвига и перестановок предлагается воспользоваться предоставленной OpenRisc возможностью расширения системы команд процессора.

В силу универсальности устройство перестановок BM_32 обеспечивает все уже реализованные в процессоре виды сдвигов, а также произвольные перестановки битов данных. Переделок и дополнений этот модуль не требует, но для его совместной работы с ЦПУ необходимы реализация новых связей, реагирование на новые сигналы, в частности, в имеющихся штатных

узлах следует провести изменения. Технически их целесообразно свести к минимуму. Для этого предлагается использовать уже реализованный и присутствующий внутри ядра OR1200 интерфейс классов дополнительных пользовательских команд, содержащих в своем названии слово cust (custom). Наиболее подходящим классом пользовательских команд, представляющим на сегодняшний день самые широкие возможности учета полезной информации для передачи ее модулю BM_32, служат команды cust5. На языке ассемблера они имеют вид

l.cust5 rD, rA, rB, L, K

Разряды машинного кода этой команды (с 31-го по 11-й) служат стандартным целям: размещению кода команды, формируемого впоследствии на линиях процессора alu_op, и указанию запрограммированных ранее регистров общего назначения. Регистр rA можно использовать для размещения входных данных. Регистр rD – для выходных данных. Регистр rB – для размещения битов маски F. Поле K (разряды 4:0 инструкции) – для указания величины сдвига, а поле L (разряды 10:5 инструкции) – для указания дополнительных кодов команд перестановок. Им соответствуют внутренние линии ЦПУ.

В проекте ORSoC процессор OR1200 уже использует 2 младших разряда поля L команд “cust5”. Они применяются для осуществления операции «вставки» младшего байта регистра rB, в качестве указателя одного из 4 мест в 32-разрядных данных регистра rA. Регистр rD в этой операции используется по своему унифицированному назначению, т. е. для записи результата.

Таким образом, разряды L[5:2] являются свободными и их можно использовать как независимо, так и в комбинации с двумя младшими разрядами. Возможность комбинированного использования следует из текста АЛУ, где в исполнении команд участвуют не отдельные разряды L, а числа, которые они образуют, распределяемые далее оператором «casez» в описании модуля.

Для наших целей, поскольку коды 0, 1, 2, 3 заняты, выбраны следующие за ними числа:

- 4 (сдвиг логический вправо (shr));
- 5 (сдвиг логический влево (shl));
- 6 (сдвиг циклический вправо (rotr));
- 7 (сдвиг циклический влево (rotl)).

Таким образом, разряд L[2] в общем формате команд может служить признаком команд «классического» сдвига.

Далее следуют команды работы с матрицей перестановок. Коды этих команд начинаются с 8, т. е. одновременно признаком их исполнения является установленный разряд L[3]:

- 8 («pex» – parallel extract);
- 9 («bsn» – baseline);
- 10 («pdep» – parallel deposit).

Модуль ЦПУ, описываемый файлом «trl» (декодер инструкций процессора), преобразует участвующие в командах поля и данные в общедоступные остальным модулям цепи с действующими сигналами на них. При этом:

– данные поля L помещаются во внутренний регистр с именем «cust5_op»;

– данные поля K – в «cust5_limm»;

– значение из регистра rA – в «operand_a»;

– значение из регистра rB – в «operand_b».

Новые связи процессора OR1200 с модулем перестановок **BM_32**

Поскольку модуль BM_32 будет управляться числами и поэтому имеет формируемые управляющие сигналы, для него необходимо сделать соответствующие цепи преобразований, на которые следует подавать сформированные процессором значения.

Адаптированный к процессору модуль перестановок в рассматриваемом техническом проекте назван именем «REP», и его текст размещается в файле or1200_rep. В целом модуль REP представляет собой «обвязку» модуля BM_32 внутренними связями ЦПУ ядра OR1200.

Исполнительная логическая часть модуля REP является продолжением АЛУ с внутренними входными портами repmr (приемник режима работы BM_32), repsr (приемник величины сдвига), repfr (приемник значения маски F), repin (приемник преобразуемых данных) и одним выходным портом repout для передачи АЛУ результата выполненного преобразования.

Логическая часть модуля, кроме операторов непрерывного присваивания (assign), никаких других операторов не содержит, что соответствует «требованиям» АЛУ. Тем не менее для изменения функции cust5 и приема результатов от модуля REP в модуле alu.v требуется провести изменения. Ввиду отсутствия лицензионного разрешения новый модуль АЛУ обозначен как alu_r.v.

Далее приводятся все остальные изменения, необходимые для обеспечения работы BM_32, реализуемые в других модулях ЦПУ процессора OR1200.

Реализация доступа к перепрограммируемой матрице перестановок

Взаимодействующей частью ЦПУ при работе модуля bm_32 является блок памяти, связанный, в свою очередь, с блоком отработки обращений к спецрегистрам. Описание этого блока является содержанием файла sprs.v. Эта часть «обвязки» модуля BM_32 отрабатывает смену условий работы модуля BM_32 с коммутационной матрицей внутри себя.

Константы управления матрицей C_1, C_2, C_3, C_4 и C_5, необходимы для задания условий работы модуля и не охвачены форматом команды cust5. В принципе они могут нести долговременный, не часто меняемый характер. Поэтому для обеспечения гибкости использования модуля BM_32 для разных промежутков времени данные условия должны храниться в регистрах.

Чтобы освободить при этом регистры общего назначения для решения задач вычислительного характера, регистры BM_32 должны быть регистрами специального назначения, расположенными, в случае встраивания модуля REP в ЦПУ OR1200, вместе с модулем BM_32. В тексте модуля указанные регистры обозначены как repc1, repc2, repc3, repc4 и repc5.

Для задания значения этим спецрегистрам, а также чтения с них информации в системе команд OR1200 имеется 2 команды l.mtcpr и l.mfspr, которые осуществляют свою работу при указании режима SM (0-й разряд) в регистре супервизора SR. Команд l.mtspr имеет формат l.mtspr rA, rB, K. По этой команде процессор должен передать содержимое регистра общего назначения rB в регистр специального назначения, адрес которого определяется операцией логического «ИЛИ» значения регистра rA и константы K, для которой в поле представления команды отведено 16 разрядов. В общем случае адрес каждого спецрегистра может быть определен множественным путем.

Аналогично для команды l.mfspr – l.mfspr rD, rA, K. По этой команде процессор должен принять и записать содержимое регистра специального назначения в регистр общего назначения rD. Адрес регистра специального назначения является результатом бинарной операции логического «ИЛИ» значения регистра rA и константы K, для которой в поле представления команды отведено 16 разрядов. В порядке дополнения отметим, что константа K в разных командах размещается в разных разрядах, а при K = 0 точный адрес спецрегистра может указываться содержимым регистра rA.

В структуре OpenRisk1000 каждой группе спецрегистров отведены свои номера от 0 до 31. В адресном поле (регистр rA) номер группы занимает места с 15 по 11 разряды, оставляя место индексу внутри себя от 0 до 1535. Это удобно ввиду организации структурно распределенных независимых звеньев как в процессоре, так и во внешних устройствах.

Первые номера групп от 0 до 11 в OR1200 уже заняты. Для дальнейшего усовершенствования процессора предназначены группы спецрегистров с номерами до 23 включительно. Поэтому для обращения к спецрегистрам можно использовать группу с номером 24 и индексом от 0 до 4, поскольку она новая и ничем иным, в том числе никакими обязательствами, не занята.

Этот номер и индекс являются определяемыми и легко сменяемыми параметрами, в связи с чем их всегда можно сменить при «перенастройке» модуля REP. Например, номер группы можно взять 0, а индекс любым числом от 1024 до 1535. Такие атрибуты приняты для обозначения регистров GPR0-GPR511 в SPR пространстве.

Декодером машинных слов, представляющих инструкции процессора OR1200, является отдельный модуль процессора, описываемый файлом «ctrl.v», на языке Verilog HDL. Выходными сигналами этого модуля, участвующими в дальнейшей работе процессора, являются:

– значение регистра rA, представленное 32-битовым сигналом «operand_a»;

– значение регистра rB, представленное 32-битовым сигналом «operand_b»;

– значение константы K, представленное 16-битовым сигналом «ex_simm»;

– 1-битовые парные сигналы «ex_spr_read» и «ex_spr_write», идентифицирующие своей совокупностью инструкции mtspr/mfspr.

При подаче этих сигналов на соответствующие входы «штатного» подмодуля процессора, описываемого файлом «sprs.v», они преобразуются в требуемые каждым сторонним модулем обработки спецсигналов следующими связными цепями:

– «spr_cs» (сигнал обращения к блоку спецрегистров);
– «spr_we» (сигнал записи в спецрегистр);
– «spr_addr» (полный адрес спецрегистра);
– «spr_dat_o» (данные, которые следует разместить в спецрегистре).

Для соблюдения принципа структурного распределения ЦПУ, для передачи данных из блока спецрегистров нового модуля REP необходимо изменить внутреннее содержимое файла «sprs.v».

Порт приема данных с регистров спецблока во внутреннем формате блока SPRS выглядят как

spr_dat_<имя блока>.

Таким образом, добавленный в SPRS входной порт должен иметь имя spr_dat_REP. Принятые по нему данные затем могут быть сняты с выходного порта блока SPRS с именем «to_wbmux» и переданы далее по цепи связи ЦПУ с именем «sprs_dataout». Ввиду отсутствия лицензионного разрешения новый файл модуля SPRS обозначен как «sprs_r.v».

Общие константы, определяющие состав процессора при компиляции проекта

Открытая архитектура процессора OR1200 предполагает выбор пользователем тех или иных возможностей скомпилированной им системы, сообразно достигаемым целям своего проекта. Этот выбор осуществляется введением в список включенных определений параметров, расположенных в файле or1200_defines.v и имеющих вид

`define OR1200_<имя блока>_IMPLEMENTED.

Например:

– `define OR1200_MULT_IMPLEMENTED;
– `define OR1200_MAC_IMPLEMENTED;
– `define OR1200_PM_IMPLEMENTED;
– `define OR1200_DU_IMPLEMENTED;

– `define OR1200_PIC_IMPLEMENTED;
– `define OR1200_TT_IMPLEMENTED.

К этому списку в рамках проекта следует добавить параметр

`define OR1200_REP_IMPLEMENTED.

Поскольку проект предполагает использование класса команд cust5, следует разблокировать АЛУ, убрав символы комментария "//" перед записью

OR1200_ALUOP_CUST5

в файле "or1200_defines.v".

Для работы с модулем REP авторами статьи составлен новый файл определений параметров "or1200_defines.v".

Макет процессора OR1200 с новыми функциями перестановки битов

Для отладки и демонстрации работоспособности построенных в проекте модулей необходимо собрать процессор OR1200 в новой конфигурации и составить демонстрационную программу.

Полный процесс компиляции такого проекта с созданием всех имитационных моделей трудоемкий и будет повторяться каждый раз при обращении к новым вариантам проекта. Поэтому в целях сокращения времени отладки и устранения возможных ошибок в проектировании процесс моделирования предлагается на первых этапах произвести на укороченном макете OR1200, имитирующем его работу только в части модуля BM_32.

Для создания и демонстрации работы такого макета участие всего процессора не требуется.

Новая тестовая программа, помимо укороченного процессора, содержит укороченные модели памяти данных DMMU в виде набора «рабочих» регистров. Память команд (модуль IMMU) и их выборку (модуль IF) имитирует одна линейная схема, звенья которой размещены последовательно и разделены элементами задержки времени.

Присутствие «укороченного» макета ЦПУ сводится к описанию необходимых связей и регистров, непосредственно задействованных портами модулей АЛУ, REP и SPRS в их совместной работе. Сами указанные модули моделируются операторами вхождения модулей alu_r, rep и sprs_r.

Для задания конкретных значений внутренним спецрегистрам модуля REP необходимо использовать команду l.mtspr rA,rB,K. В отличие от команды чтения (l.mtspr rA,rB,K) данная команда исполняется процессором в подмодуле SPRS при установленном режиме супервизора, который устанавливается разрядом SM (0-й разряд) специального регистра SR (17-й в группе системных регистров с номером 0).

В построенной демонстрационной модели для задания команд записи и чтения используются следующие эквивалентные им цепочки (физических) операторов.

Для команды l.mtspr эквивалентом в укороченном нами ядре ЦПУ является цепочка:

```
operand_a = 0; // ук-ть базовый адрес
operand_b      =      ...; // ук-ть записываемые данные
ex_simm = 0;//сформировать поле команды, 32 разряда
   #1; // время, необходимое для восприятия смены адреса
ex_simm[`OR1200_SPR_GROUP_BITS] = ...; //ук-ть № группы
ex_simm = ex_simm | i; //ук-ть i-й регистр
   ex_spr_read        =      0; //ук-ть ком. записи
   ex_spr_write       =      1;
```

Для команды «l.mfspr» эквивалентом в укороченном нами ядре ЦПУ является цепочка:

```
operand_a = 0; // ук-ть базовый адрес
ex_simm = 0;//ук-ть 32 разряда
   #1; // для восприятия смены адреса
ex_simm[`OR1200_SPR_GROUP_BITS] = ...; //ук-ть № группы
ex_simm = ex_simm | i; //ук-ть i-й регистр
   ex_spr_read        =      0; //ук-ть ком. чтения
   ex_spr_write       =      1;
   #1;
   ...      =      spr_dat_rep; // указать место записи считывания данных
```

Для наглядной демонстрации команды сдвига обновленного процессора на Verilog HDL была написана и совместно с обновленным процессором OR1200 синтезирована в единую электрическую схему специальная тестовая программа. В программе Xilinx ISE проект, в котором синтезируется схема, назван Test_REPLACE.

Тексты модели ориентированы на работу в симуляторе ISim от XILINX [13]. Программа ISim позволяет просматривать эпюры всех сигналов на всех линиях отстраиваемого макета, а также следить за всеми целевыми параметрами и результатами работы исполняемых действий в специальном рабочем окне, называемом «console». Для вывода данных в это окно в тексте отстраиваемой модели использованы операторы вывода «$write» .

Отметим, что для иллюстрации работы новых регистров памяти C_i в них записываются значения тестовой переменной CCC. Для правильной работы процессора и полноты контроля в демонстрационную модель заведен сигнал rst («сброс»), применяемый в начале работы схемы.

Ниже приводятся выборочные фрагменты результатов работы схемы после исполнения процедуры «initial», взятые в рабочем окне стендовой

программы «ISim». При этом 32-разрядная маска F в этом фрагменте попеременно принимает значения –1 и 0. Число сдвигов указанной операции следует за знаком «#» и указано в десятиричном исчислении.

```
Simulator is doing circuit initialization process.
Finished circuit initialization process.
      C( 0 ) = 00000000  C( 1 ) = 00000000  C( 2 ) = 00000000  C( 3 ) = 00000000
      C( 4 ) = 00000000
      F  = FFFFFFFF       shr #  0
       31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3
2  1  0

      C( 0 ) = 00000001  C( 1 ) = 00000001  C( 2 ) = 00000001  C( 3 ) = 00000001
      C( 4 ) = 00000001
      F  = 00000000       shr #  1
        0 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4
3  2  1

      C( 0 ) = 00000002  C( 1 ) = 00000002  C( 2 ) = 00000002  C( 3 ) = 00000002
      C( 4 ) = 00000002
      F  = FFFFFFFF       shr #  2
        0  0 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5
4  3  2

      C( 0 ) = 00000003  C( 1 ) = 00000003  C( 2 ) = 00000003  C( 3 ) = 00000003
      C( 4 ) = 00000003
      F  = 00000000       shr #  3
        0  0  0 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6
5  4  3
. . .
. . .
      C( 0 ) = 00000020  C( 1 ) = 00000020  C( 2 ) = 00000020  C( 3 ) = 00000020
      C( 4 ) = 00000020
      F  = FFFFFFFF       shl #  0
       31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3
2  1  0

      C( 0 ) = 00000021  C( 1 ) = 00000021  C( 2 ) = 00000021  C( 3 ) = 00000021
      C( 4 ) = 00000021
      F  = 00000000       shl #  1
       30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2
1  0  0
```

Приведенные результаты испытаний демонстрационной модели показывают правильность работы модуля BM_32 в составе процессора OR1200 и сборки данного процессора с новыми модулями. При этом можно отметить, что простые команды сдвига от управляющих параметров C_i и F не зависят.

Таким образом, в статье описан опыт работы авторов по созданию синтезируемых процессорных систем на базе семейства OpenRisc 1000. Для разработки и моделирования использовалась программа Xilinx ISE.

Приведены подробное описание и особенности разработки 32-разрядного RISC-микропроцессора с улучшенной манипуляцией битами данных. Новые команды микропроцессора позволяют увеличить его производительность и сократить энергопотребление от 2 до 10 раз при решении задач обработки изображений, распознавания образов, кодирования, стеганографии, криптографии, обработки сигналов и т. п.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Сотов Л. С.* Об эффективности использования специальных команд преобразования форматов данных в вычислительной технике // Гетеромагнитная микроэлектроника : сб. науч. тр. Саратов : Изд-во Сарат. ун-та, 2011. Вып. 10 : Гетеромагнитная микро- и наноэлектроника. Прикладные аспекты. Экономика. Методические аспекты физического образования. С. 61–80.

2. *Сотов Л. С.* Методы синтеза устройств, выполняющих инструкции перестановки битов данных // Гетеромагнитная микроэлектроника : сб. науч. тр. Саратов : Изд-во Сарат. ун-та, 2011. Вып. 10 : Гетеромагнитная микро- и наноэлектроника. Прикладные аспекты. Экономика. Методические аспекты физического образования. С. 25–50.

3. Пат. 2320000 Российская Федерация, МПК G0 6F 7/76, G0 6F 12/14. Дешифратор управляемой побитовой транспозиции информации, хранимой в персональной ЭВМ / заявители Молодченко Ж. А., Сотов Л. С., Харин В. Н. ; патентообладатель Сарат. гос. ун-т им. Н. Г. Чернышевского. – № 2007105175/09 ; заявл. 13.02.2007 ; опубл. 20.03.2008, Бюл. № 8. – 6 с.

4. Пат. 2390052 Российская Федерация, МПК G0 6F 7/76. Дешифратор управляемой перестановки информации, хранимой в персональной ЭВМ / заявители Молодченко Ж. А., Сотов Л. С., Харин В. Н. ; патентообладатель Сарат. гос. ун-т им. Н. Г. Чернышевского. – № 2008132009/09 ; заявл. 06.08.2008 ; опубл. 20.05.2010, Бюл. № 14. – 8 с.

5. Пат. 2390049 Российская Федерация, МПК G0 6F7/00. Параллельный дешифратор управляемой транспозиции нформации, хранимой в персональной ЭВМ / заявители Молодченко Ж. А., Сотов Л. С., Харин В. Н. ; патентообладатель Сарат. гос. ун-т им. Н. Г. Чернышевского. – № 2008139529/09 ; заявл. 07.10.2008 ; опубл. 20.05.2010, Бюл. № 1. – 8 с.

6. *Hilewitz Y., Shi Z. J., Lee R. B.* Comparing Fast Implementations of Bit Permutation Instruction // IEEE, Signals. Systems and Computers. Vol. 2. P. 1856–1863.

7. *Hilewitz Y., Lee R.* A New Basis for Shifters in General-Purpose Processors for Existing and Advanced Bit Manipulations // IEEE Transactions on Computing. 2009. Vol. 58, № 8. P. 1035–1048.

8. *Молодченко Ж. А., Сотов Л. С., Харин В. Н.* Математические модели транспозиционных преобразований // Информационно-измерительные и управляющие системы. 2007. Т. 5, № 12. С. 58–60.

9. *Соболев С. С., Харин В. Н., Сотов Л. С.* Модели устройств кросс-кластерных перестановок данных в ЭВМ // Вестн. компьютерных и информационных технологий. 2009. № 12. С. 51–55.

10. *Молодченко Ж. А., Сотов Л. С., Харин В. Н.* Модели аппаратных функциональных формирователей перестановок // Информационно-измерительные и управляющие системы. 2009. Т. 7, № 10. С. 78–84.

11. *Сотов Л. С.* Комбинаторная модель функционального формирователя разбиений бинарного множества // Информационные технологии. 2010. № 10. С. 46–52.

12. Пат. 2488161 Российская Федерация, МПК G0 6F 11/00. Устройство перестановок и сдвигов битов данных в микропроцессорах / заявитель Сотов Л. С. – № 2011145864/08 ; заявл. 14.11.2011 ; опубл. 20.07.2013, Бюл. № 20. – 27 с.

13. Xilix ISE Web Pack. URL : www.xilinx.com (дата обращения : 01.10.2014).

14. *Сотов Л. С., Ачкасов В. Н.* Универсальный модуль манипуляции битами данных в микропроцессорах // Гетеромагнитная микроэлектроника : сб. науч. тр. Саратов : Изд-во Сарат. ун-та, 2011. Вып. 11 : Гетеромагнитная микро- и наноэлектроника. Прикладные аспекты. Экономика. Методические аспекты физического образования. С. 57–73.

15. *Молодченко Ж. А., Харин В. Н., Сотов Л. С.* Алгоритм создания диверсификационного метода битовых преобразований // Естественные и технические науки. 2007. № 6. С. 222–225.

16. *Молодченко Ж. А., Харин В. Н., Овчинников С. В., Сотов Л. С.* Модели аппаратных акселераторов перестановок бинарных множеств // Гетеромагнитная микроэлектроника : сб. науч. тр. Саратов : Изд-во Сарат. ун-та, 2008. Вып. 4 : Гетеромагнитная микро- и наноэлектроника. Прикладные аспекты. Устройства различного назначения. С. 11–23.

17. *Молодченко Ж. А., Сотов Л. С., Харин В. Н.* Моделирование архитектуры акселератора битовых перестановок с использованием САПР SYSTEM STUDIO фирмы SYNOPSYS // Гетеромагнитная микроэлектроника : сб. науч. тр. Саратов : Изд-во Сарат. ун-та, 2008. Вып. 3 : Гетеромагнитная микро- и наноэлектроника. Прикладные аспекты. С. 60–66.

18. *Ляшенко А. В., Сотов Л. С.* Стохастические генераторы упорядоченных разбиений конечных множеств с быстрым ростом энтропии // Гетеромагнитная микроэлектроника : сб. науч. тр. Саратов : Изд-во Сарат. ун-та, 2010. Вып. 8 : Гетеромагнитная микро- и наноэлектроника. Системы информационной безопасности. Прикладные аспекты. С. 57–72.

19. *Сотов Л. С., Солопов А. А., Фарафонова А. В.* Модель инволютивного транспозиционного преобразователя // Гетеромагнитная микроэлектроника. Саратов : Изд-во Сарат. ун-та, 2010. Вып. 8 : Гетеромагнитная микро- и наноэлектроника. Системы информационной безопасности. Прикладные аспекты. С. 34–46.

20. *Сотов Л. С., Хвалин А. Л.* Средства разработки и исследования архитектурных моделей в сапр System Studio. Ч. 1 : Использование инструментов system studio при моделировании матричного генератора перестановок // Гетеромагнитная микроэлектроника : сб. науч. тр. Саратов : Изд-во Сарат. ун-та, 2008. Вып. 5 : Прикладные аспекты микро- и наноэлектроники. С. 121–145.

21. *Сотов Л. С., Хвалин А. Л.* Средства разработки и исследования архитектурных моделей в САПР System Studio. Ч. 2 : Основные объекты SYSTEMC и их использование // Гетеромагнитная микроэлектроника : сб. науч.тр. Саратов : Изд-во Сарат. ун-та, 2008. Вып. 5 : Прикладные аспекты микро- и наноэлектроники. С. 146–176.

22. *Сотов Л. С.*, Соболев С. С., *Харин В. Н.* Кросс-кластерная коммутационная матрица для аппаратной поддержки управляемой перестановки данных в криптографических системах // Проблемы информационной безопасности. Компьютерные системы. 2009. № 4. С. 56–63.

23. *Молодченко Ж. А., Сотов Л. С., Харин В. Н.* Аппаратный акселератор сервера форматирования данных // Надежность и качество : тр. междунар. симпозиума. 2007. Т. 1. С. 134–136.

24. *Назаров С. И., Сотов Л. С., Ляшенко А. В.* Процессор с улучшенной манипуляцией битами данных для средств навигации, обработки сигналов и изображений, криптографии, мобильных диагностических устройств // Гетеромагнитная микроэлектроника : сб. науч. тр. Саратов : Изд-во Сарат. ун-та, 2014. Вып. 16 :

Гетеромагнитная микро- и наноэлектроника. Методические аспекты физического образования. Экономика в промышленности. С. 51–63.

25. OpenRisc processor. URL : http://opencores.org (дата обращения : 01.10.2014).