УДК 004.318

ОПЫТ РАЗРАБОТКИ САМОДЕЛЬНОГО ПРОЦЕССОРА НА ПЛИС

Кулагин О.В.
ФГУП «Пензенский научно-исследовательский электротехнический институт», г. Пенза,
тел. 8-(937)-400-11-38 (г. Пенза), e-mail: ruz1967@yandex.ru

Аннотация. В статье представлен созданный автором на ПЛИС самодельный оригинальный 32-разрядный стековый процессор, и изложена методика его проектирования. Предполагается, что она может быть полезной разработчикам специализированных цифровых устройств.
Ключевые слова: Процессор, команда, программа, стек, память.

Annotation. In the article the self-made original 32-bit stack processor and the method of his designing are presented. It is supposed, that the method can be useful for developers of the specialized digital devices.
Keywords: processor, command, program, stack, memory, PLD.

Введение. Возможности современной микроэлектроники, и прежде всего характеристики программируемых логических интегральных схем (ПЛИС), переводят процесс проектирования процессоров из числа уникальных научно-технических проблем в число общедоступных задач. В статьях [1, 2, 3, 4] по данной теме приводятся следующие основные причины, делающие проектирование небольших самодельных процессоров на ПЛИС (называемых еще софт-процессорами) выгодными:

1. Конечный автомат с числом состояний 50 и выше проще отлаживать и сопровождать в программном, а не аппаратном варианте.
2. При реализации узкоспециализированных алгоритмов аппаратное выполнение специализированных операций позволит сократить объем программного кода.

Следовательно, в каждом самодельном процессоре должен реализовываться оригинальный набор команд, определяемый потребностями алгоритма, реализуемого на таком процессоре. На основе этого набора необходимо создать: 1) сам процессор в виде проекта для ПЛИС и 2) средства программирования для процессора. Начинать проектирование процессора необходимо с определения назначения его команд без учета их кодировок.

Затем необходимо выбрать архитектуру создаваемого процессора. Существуют два основных типа процессорных архитектур – регистровая и стековая. Первая предусматривает наличие в процессоре определенного набора регистров (как правило поделенных на отдельные поля) для хранения операндов, к которым организован произвольный доступ. Вторая основана на использовании стека – структуры памяти, организованной по принципу «первым вошел – последним вышел».

Проверка работоспособности нового процессора возможна только с программным кодом для него. Для предварительной проверки этого кода без использования процессора необходимо создать программный симулятор разрабатываемого процессора и в нем проверить правильность функционирования программного кода для него.

В данной статье представлен созданный автором на ПЛИС самодельный оригинальный 32-разрядный стековый процессор, и на примере его разработки наглядно показана методика его проектирования.

Назначение созданного процессора. Быстрая смена поколений процессоров и большой объем отлаженного исполняемого кода для старых процессоров заставляет обратить внимание на проектирование «систем на кристалле» (СНК) с самодельными процессорами. Такие СНК должны быть дополнены средствами перекомпиляции исполняемых модулей программ в код для самодельных процессоров. В настоящее время автором завершаются работы по созданию таких средств на основе собственного оригинального дизассемблера Диздек [6]. Он уже настроен на дизассемблирование исполняемых модулей программ процессо-

ра TMS320C50 фирмы Texas Instruments и их последующую компиляцию на язык Паскаль в версии системы программирования Delphi 6.0, и в машинный код созданного процессора, который, таким образом, может быть использован вместо процессора TMS320C50.

Поэтому, для созданного процессора была разработана система команд, включающая помимо общеупотребительных операций две операции, используемые в сигнальных процессорах. Первая из этих команд – операция знакорасширения – предполагает распространение старшей значащей единицы слова во все предшествующие ей старшие незначащие биты этого слова. Вторая команда названа операцией «переворот». Она предполагает перестановку младших $\boldsymbol{m}$ бит в слове $\boldsymbol{A}$ по правилу $a_i = a_{m\text{-}i\text{-}1} \mid i \le m$. Например, результатом «переворота» пяти младших бит в байте $a_7a_6a_5a_4a_3a_2a_1a_0$ будет число $a_7a_6a_5a_0a_1a_2a_3a_4$.

Созданный процессор построен по стековой архитектуре и реализован на макетной плате фирмы Altera с ПЛИС EP3C120F780C7N (серия Cyclone III). Основные характеристики процессора следующие:

- разрядность шины данных – 32;
- разрядность шины адресов – 21 (соответствует разрядности внешнего ОЗУ на макетной плате);
- тактовая частота – 12,5 МГц (была ограничена быстродействием внешнего ОЗУ на макетной плате, имеющего время выборки 70 нс., с внутренним ОЗУ тактовая частота процессора может быть увеличена);
- приблизительное число вентилей ПЛИС, занимаемых данным процессором – 1.400.000;
- количество команд – 48, из них 16 управляющих и 32 линейных (это команды: арифметические, логические, сравнения, обмена с памятью, записи вспомогательных регистров);
- включает полную аппаратную поддержку вложенных циклов со счетчиком, избавляющую программиста от необходимости хранить в памяти любые переменные, определяющие параметры таких циклов;
- максимальное число аппаратно поддерживаемых вложенных циклов со счетчиком – 16;
- встроенные средства отладки программ – отсутствуют (предполагается ввести их потом).

Работа процессора проверена на программе тестирования внешнего ОЗУ с искусственной имитацией несравнений считываемых из ОЗУ тестовых значений или без такой имитации. В первом случае выдавался отрицательный результат тестирования, во втором – положительный.

В табл. 1 представлены сравнительные характеристики созданного процессора и процессоров, серийно выпускаемых ОАО «Ангстрем» (Москва-Зеленоград). Из таблицы видно, что созданный процессор превосходит серийно выпускаемый 32-разрядный 1839 ВМ1 по тактовой частоте и по компактности системы команд.

Таблица 1 – Сравнительные характеристики созданного процессора и процессоров ОАО «Ангстрем».

Процессоры	**Созданный**	**КН1871ВЕ1**	**КР1878ВЕ1**	**1806ВМ2**	**1839ВМ1**
Разрядность	32	4	8	16	32
Тактовая частота, МГц	12,5	4	8	5	10
Число команд	48	39	52	77	304

Кодировка команд во многом зависит от архитектуры процессора. Статьи [1, 2, 3, 4] фактически ориентируют разработчиков на регистровую архитектуру, хотя в [3, 4] также представлена и стековая архитектура. Однако, регистровая архитектура имеет ряд существенных недостатков. Ей присущ большой набор внутренних регистров процессора, подавляющая часть которых к тому же делится на поля с отдельным программным доступом к каждому из них. Это сильно затрудняет освоение и эффективное использование нового процессора и вынуждает вводить большое число команд (или разновидностей команд) для пере-

сылок информации между отдельными комбинациями регистрами. А каждая новая команда влечет увеличение аппаратной части процессора.

В [5] показано преимущество стековой архитектуры перед регистровой. Использование стековой архитектуры для построения процессора влечет: 1) использование для его программирования языка, основанного на польской инверсной записи (ПОЛИЗ), и 2) организацию исполняемых модулей программ для него в виде сшитого кода (т.е. использования команд вызовов подпрограмм вместо команд адресных переходов). Стековая архитектура упрощает как сам процессор, так и его программирование, и по этой причине созданный процессор построен по данной архитектуре. Однако, в созданном процессоре сохранены команды адресных переходов для улучшения читаемости программы.

В процессоре содержатся 20-уровневые 32-разрядные стеки: 1) адресов, и 2) арифметический. Кроме того, имеется полная аппаратная поддержка вложенных циклов со счетчиком на основе трех 16-уровневых 32-разрядных стеков: 1) начальных значений циклов **C0**, 2) конечных значений циклов **СК**, 3) изменений счетчика циклов **CD**, и 3) счетчиков циклов **ind.**

В любой программе необходимо анализировать те или иные условия, и общепринятой практикой стала организация в процессорах определенных наборов триггеров-флажков, автоматически устанавливаемых в требуемое значение после каждой арифметической и логической операции. Однако, в созданном процессоре принят совершенно другой подход. В нем имеется один-единственный программно устанавливаемый флаг, названный **USL,** по которому выполняются все условные переходы и условные вызовы подпрограмм.

На первый взгляд кажется, что аппаратное формирование флагов после каждой команды упрощает программирование. Однако, не слишком заметная экономия кода при программировании оборачивается разрастанием дизассемблируемого кода, в котором в каждой дизассемблируемой команде необходимо программно прописывать установку флагов. Этого не произойдет при предлагаемом подходе, когда в программе будет явно указано условие перехода. Это упростит дизассемблирование и декомпиляцию программ, а также облегчит их переносимость на другие платформы.

Для циклов с вычисляемым в самой программе верхним пределом используется специальный регистр **shd**. Вместе с флагом **USL** эти регистры названы *вспомогательными*. Они загружается из вершины стека специальной командой.

Полная аппаратная поддержка вложенных циклов со счетчиком выражается в том, что начальные, конечные значения вложенных циклов, а также счетчики циклов и их приращения, выделены в отдельные стеки. Предусмотрена загрузка счетчика цикла в стек и его аппаратное изменение в конце цикла. Для его аппаратного изменения предусмотрен стек **CD**, уровни которого имеют структуру, представленную в табл. 2.

Таблица 2 – Структура уровня стека CD.

Биты	**31…24**	**23**	**22…0**
Назначение их	0	1 – отнять приращение от счетчика, 0 – прибавить приращение к счетчику	Приращение счетчика цикла (в 23 битах)

При появлении вложенного цикла специальной командой осуществляется переход на следующий уровень стеков **C0**, **СК**, **CD**, **ind**. По окончании цикла со счетчиком в случае, если использовались регистры не самых нижних уровней в стеках, происходит автоматический возврат на предыдущий уровень данных стеков.

Такая аппаратная поддержка вложенных циклов со счетчиком полностью избавляет программиста от необходимости хранения во внешней памяти переменных, определяющих параметры этих циклов.

Система команд. Включает 48 команд, из них 16 управляющих и 32 линейные (команды: арифметические, логические, сравнения, обмена с памятью, записи вспомогательных регистров).

Каждая управляющая команда представляет собой 32-разрядное слово, старшие четыре бита в котором определяют код операции (КОП), а остальные биты представляют собой поле данных. Схематично это представлено в табл. 3.

Таблица 3 – Кодировка управляющих команд процессора.

Коды КОП		**Назначение команд**	**Поле данных в команде**
10	**16**		
0	0h	Указатель начала последовательности линейных команд	*Произвольное значение*
1	1h	Безусловный адресный переход	Адрес перехода
2	2h	Операции с регистрами циклов	*см. табл. 4*
3	3h	Запись начального значения счетчика цикла	Записываемое значение
4	4h	Запись конечного значения счетчика цикла	Записываемое значение
5	5h	Возврат из подпрограммы	Нули
6	6h	Адресный переход при USL=0	Адрес перехода
7	7h	Адресный переход при USL=1	Адрес перехода
8	8h	Цикл со счетчиком, в котором верхний предел – константа	Адрес перехода на первую команду тела цикла[1]
9	9h	Цикл со счетчиком, в котором верхний предел считывается с регистра **shd**	Адрес перехода на первую команду тела цикла[1]
10	Ah	Цикл с постусловием	Адрес перехода на первую команду тела цикла[1]
11	Bh	Конец программы	Нули
12	Ch	Сохранение адреса в подвершине стека	Сохраняемое значение
13	Dh	Вызов подпрограммы	Адрес подпрограммы
14	Eh	Вызов подпрограммы при USL=0	Адрес подпрограммы
15	Fh	Вызов подпрограммы при USL=1	Адрес подпрограммы

Таблица 4 – Структура команд, определяющих операции с регистрами циклов.

Биты регистров	**31…28**	**27…25**	**24**	**23**	**22…8**	**7…4**	**3…0**
Запись счетчика цикла **ind** в стек	2h	0h		0h		Число α	2h
Запись стека изменения счетчика цикла **CD**	2h	0h	1	1 – отнять приращение от счетчика, 0 – прибавить приращение к счетчику	Приращение счетчика цикла		
Установка счетчика цикла **ind** в начальное значение из текущего уровня стека **C0**	2h	0h		0h			1h
Начало вложенного цикла[2]	2h	0h		0h			3h

Число α отнимается от номера текущего уровня стека **ind**, определяя таким образом требуемую ячейку для считывания значения. Текущему значению стека **ind** соответствует α=0, предыдущему – α=1, и т.д..

---

[1] Последняя команда тела цикла находится перед этой командой.

[2] Должна предварять все команды, к этому циклу относящиеся.

Линейные команды представлены в табл. 5, где через B обозначена подвершина стека, а через D – вершина стека.

Последовательности линейных команд. Последовательности линейных команд должны быть оформлены в виде подпрограмм, структура которых представлена в табл. 6.

Таблица 5 – Кодировки линейных команд.

Коды		Команды	Обозначения операций	
10	**16**		**Русские**	**Английские**
0	00h	Пустая операция НОП	НОП	NOP
1	01h	Инверсия	~	~
2	02h	Операция .orb.	ИЛИ1	.orb.
3	03h	Операция .andb.	И1	.andb.
4	04h	Операция .xorb.	МОД2В1	.xorb.
5	05h	Сложение	+	+
6	06h	Вычитание B D - (в дополнительном коде)	–	–
7	07h	Умножение	*	*
8	08h	Деление B D /	/	/
9	09h	Взятие остатка B D .mod.	МОДУЛЬ	.mod.
10	0Ah	Дизъюнкция	ИЛИ	.or.
11	0Bh	Конъюнкция	&	&
12	0Ch	Сложение по модулю два	МОД2	.xor.
13	0Dh	Сравнение B≠D	.<>.	.<>.
14	0Eh	Сравнение B>D	>	>
15	0Fh	Сравнение B≥D	.>=.	.>=.
16	10h	Сравнение B<D	<	<
17	11h	Сравнение B≤D	.<=.	.<=.
18	12h	Сравнение B=D	.=.	.=.
19	13h	Операция «Мультиплексор» – выделить в числе B бит D: B D .MS.	МС	.MS.
20	14h	Операция B D .max.	МАКС	.max.
21	15h	Операция B D .min.	МИН	.min.
22	16h	Сдвиг влево	СДВИГЛ	.shl.
23	17h	Сдвиг вправо	СДВИГП	.shr.
24	18h	Циклический сдвиг (B D .rol.)	ЦСДВИГЛ	.rol.
25	19h	Циклический сдвиг (B D .ror.)	ЦСДВИГП	.ror.
26	1Ah	Знакорасширение старших B разрядов числа D	ЗНАКРАС	_znakras
27	1Bh	Операция «переворот» - в слове B перевернуть D младших разрядов	_переворот	_переворот
28	1Ch	Вершину стека выдать в управляющий автомат	каЗАП	_kaWR
29	1Dh	Запись условного бита (Копирование младшего бита выхода АЛУ в триггер условия USL)	услЗАП	_uslWR
30	1Eh	Запись в память (B D _zuWR, где B – данные, а D – адрес)	зуЗАП	_zuWR
31	1Fh	Чтение из памяти (D _zuRD, где D – данные)	зуЧТ	_zuRD

Каждая подпрограмма, содержащая линейные команды, предваряется и завершается специальными командами, присутствующими в каждой такой подпрограмме. Слова в этих подпрограммах организованы в блоки линейных команд, каждый из которых содержит слово признаков, структура которых представлена в табл. 7.

В последовательностях линейных команд могут встречаться слова данных трех видов:

константы, линейные команды, управляющие команды. Признаки слов как раз и обозначают эти разновидности слов данных. Кодировка признаков представлена в табл. 8.

Симулятор процессора. Для отладки программ данного процессора был создан его программный симулятор, на котором была проверена работоспособность двух перекомпилированных программ. Они работали идентично исходным.

Таблица 6 – Подпрограммы – последовательности линейных команд.

<table>
<tr><td colspan="2">Управляющая команда – указатель начала последовательности линейных команд (код 0xxxxxxxh)</td></tr>
<tr><td>Слово признаков 1 (см. табл. 7)</td><td rowspan="17">Блок линейных команд 1
(включает 16 слов данных)</td></tr>
<tr><td>Слово данных 1</td></tr>
<tr><td>Слово данных 2</td></tr>
<tr><td>Слово данных 3</td></tr>
<tr><td>Слово данных 4</td></tr>
<tr><td>Слово данных 5</td></tr>
<tr><td>Слово данных 6</td></tr>
<tr><td>Слово данных 7</td></tr>
<tr><td>Слово данных 8</td></tr>
<tr><td>Слово данных 9</td></tr>
<tr><td>Слово данных 10</td></tr>
<tr><td>Слово данных 11</td></tr>
<tr><td>Слово данных 12</td></tr>
<tr><td>Слово данных 13</td></tr>
<tr><td>Слово данных 14</td></tr>
<tr><td>Слово данных 15</td></tr>
<tr><td>Слово данных 16</td></tr>
<tr><td colspan="2">Блок линейных команд 2 (включает 16 слов данных)</td></tr>
<tr><td colspan="2">: : : : : : : : :</td></tr>
<tr><td colspan="2">Блок линейных команд m (включает от 1 до 16 слов данных)</td></tr>
<tr><td colspan="2">Управляющая команда – возврат из подпрограммы (код 50000000h)</td></tr>
</table>

Таблица 7 – Кодировка слова признаков.

Биты слова признаков	**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**
Номер слова данных, к которому относится признак в этих битах	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

Таблица 8 – Кодировка признаков.

Разновидности слов данных	**Двоичная кодировка признаков**
Код линейной операции	00
Константа	01
Управляющая команда	10

	11

Создание программного симулятора будущего процессора является необходимым этапом процесса создания этого процессора. Симулятор позволит проверить правильность:

1. Технических решений, реализованных в процессоре.
2. Первоначально отладить систему команд процессора.
3. Отладить программу для данного процессора.

Иными словами, симулятор позволяет получить отлаженный код для еще не отлаженного процессора, который в дальнейшем позволит отладить и сам процессор.

Модель памяти, используемая для отображения регистров. Адресное пространство памяти созданного процессора разделено на изолированные страницы, обозначаемые согласно табл. 9.

Таблица 9 – Страницы памяти.

Номер страницы памяти	Обозначение страницы	Назначение страницы
0.	PROG	Коды команд.
1.	AP	Адресная информация для кодов команд.
2.	KP	Константы для кодов команд.
3.	PRER	Прерывания.
4.	REGM	Внутренние регистры процессора, разрядностью не выше шины данных.
5.	DAN	Память данных.
6.	PRT	Область ввода-вывода (порты).
7.	REGB	Внутренние регистры процессора, разрядностью больше шины данных (32 бита).

В программе адрес любой ячейки памяти A определяется согласно формулы $A = N * 10000000h + a$, где N – номер страницы памяти (N = 1?7), а a – адрес ячейки внутри данной страницы.

Вывод. В статье представлен оригинальный 32-разрядный стековый процессор, реализованный на ПЛИС, и изложена методика его проектирования.

Литература.

1. Каршенбойм И. Микропроцессор своими руками-3. Ассемблер и софт-симулятор. // Компоненты и технологии, 2006, № 3. – с.154–159.
2. Каршенбойм И. Микропроцессор своими руками-3. Ассемблер и софт-симулятор. // Компоненты и технологии, 2006, № 4. – с.200–204.
3. Тарасов И. Проектирование конфигурируемых процессоров на базе ПЛИС. // Компоненты и технологии, 2006, № 2. – с. 78–83.
4. Тарасов И. Проектирование конфигурируемых процессоров на базе ПЛИС. // Компоненты и технологии, 2006, № 4. – с. 68–73.
5. Сидоров С.А., Шумаков М.Н. ДССП как открытая система. // URL: http://kis.pcweek.ru/kis/win/techno/dssp.html.
6. Кулагин О.В. Диздек – дизассемблер с элементами декомпиляции. // URL: http://www.autotech.mrsu.ru/2009/Dizdek.pdf.