Новая микроконтроллерная архитектура для космических применений

Игорь ПОТАПОВ
pip@niiet.ru
Владимир СМЕРЕК
smerek@niiet.ru

Сегодня на рынке микроконтроллеров господствует два типа архитектур: RISC и CISC, причем RISC-микроконтроллеры в последнее время заметно потеснили CISC. Для того чтобы понять, почему это произошло, рассмотрим особенности обоих подходов к созданию микроконтроллерных архитектур.

Архитектуры CISC (complex instruction set computer — компьютер с полным набором команд) первыми появились на рынке и имеют следующие особенности:

1. Арифметические команды могут выполнять действия с большим набором регистров, расположенных в области ОЗУ.
2. Поддержка всех типов адресации арифметическими инструкциями.
3. Нефиксированная длина команд.

В отличие от CISC, RISC (restricted (reduced) instruction set computer — компьютер с сокращенным набором команд) — это архитектура процессора, в котором быстродействие увеличивается за счет упрощения инструкций, чтобы их декодирование было более простым, а время выполнения — меньшим. В современных RISC-архитектурах большинство инструкций выполняются за один такт. Чтобы достичь такой производительности, вводятся следующие ограничения:

1. Для обеспечения максимальной производительности подсистемы выборки команд фиксируется длина инструкций и вводятся ограничения на выравнивание адресов инструкций. Это позволяет, выбрав команды из определенной области памяти, за один такт определить все находящиеся в данной области команды. Благодаря чему также упрощается наращивание разрядности шины, по которой выбирают инструкции, при этом производительность подсистемы выборки команд растет прямо пропорционально увеличению разрядности. Например, при установлении длины инструкции в 32 бит, выбрав из области памяти 128 бит информации, можно за один такт определить все 4 инструкции. В случае нефиксированной длины инструкции ядру может потребоваться от 1 до 16 тактов (при условии, что длина команд составляет от 8 до 128 бит).
2. Выполнение арифметических инструкций только со специальной областью регистров, не находящихся в ОЗУ. Это позволяет выполнять подобные инструкции за один такт. В случае применения ОЗУ в арифметических операциях, как происходит в CISC-архитектурах, минимальное количество тактов будет 4 (1 такт — выборка первого операнда, 2-й — выборка второго, 3-й — сама арифметическая операция в АЛУ, 4-й — сохранение результата). При использовании в CISC-архитектурах двухпортовых блоков ОЗУ возможно ускорение CISC-операций на один такт благодаря одновременной выборке операндов.

Развитию RICS-архитектур способствует несколько факторов:

1. Снижение стоимости внутрикристальной flash, благодаря чему можно использовать команды фиксированной длины без учета избыточности (не все биты команд могут применяться для кодирования операции) и генерировать лишний код для обмена данными с регистрами, с которыми выполняются арифметические операции.
2. Возможность решения задач пользовательских вычислений с помощью конечного числа регистров.
3. Рост вычислительных возможностей интегрированных средств разработки, вследствие чего при вышеописанных ограничениях пользовательские программы удается оптимизировать по скорости выполнения или объему кода, что важнее. Происходит перенос части работы с ядра на ПК. Определяющей производительность становится связка «компилятор — ядро».

В производственной линейке ОАО «НИИЭТ» есть изделия, основанные на том или ином описанном выше подходе. Но для нового микроконтроллерного ядра, которое будет составлять основу ИМС «Обработка-28», решено совместить достоинства CISC- и RISC-архитектур и постараться избежать следующих их недостатков:

1. Недостаток CISC-архитектур: для всех арифметических операций необходимо тратить время на выборку операндов и сохранение результатов в медленном ОЗУ, хотя в большинстве пользовательских алгоритмов в вычислениях применяется только конечное число регистров.
2. Недостаток RISC-архитектур: рост объема кода программы. Если в CISC-архитектуре используется одна инструкция для сложения данных, находящихся в двух ячейках ОЗУ, то в RISC-архитектуре в общем случае потребуется четыре инструкции при возможной избыточности каждой из-за требований фиксированности длины.

История создания микроконтроллерных CISC-архитектур в ОАО «НИИЭТ»

Исторически ОАО «НИИЭТ» развивало несколько направлений CISC-архитектур.

1. MCS-51. Лежит в основе следующих изделий: Н1830ВЕ31, Н1830ВЕ51, 1830ВЕ32У, 1882ВЕ53У, 1882ВМ1Т. Особенности последних трех схем — использование модели ядра, созданной собственными силами в ОАО «НИИЭТ».
2. MCS-96. Лежит в основе следующих изделий: Л1874ВЕ36, 1874ВЕ36, 1874ВЕ06Т, 1874ВЕ76Т, 1874ВЕ16Т, 1874ВЕ86Т, 1874ВЕ66Т, 1874ВЕ05Т, 1874ВЕ96Т, 1874ВЕ7Т. В рамках разработки последних двух ИМС были проведены работы по созданию Verilog-модели ядра, которое имело ряд особенностей, позволяющих говорить о создании собственной масштабируемой CISC-платформы, способной гибко наращивать вычислительные возможности ядер. К особенностям подхода, выбранного для создания модификации ядра, получившей впоследствии название AMCS-96 (Advance MCS-96), относятся:

- Создание подсистемы выборки команд, максимально независимой от подсистемы выполнения команд при возможности гибкой настройки длины каждой из инструкций. Блок информации, содержащий команды (при максимальной длине команды до 7 байт блок может содержать или неполную команду, или до четырех 8-битных команд), по 32-разрядной шине команд поступает в блок очереди команд, который автоматически на основании данных декодера длины инструкций помещает команды в очередь. Выборка команд происходит асинхронно от выполнения команд, сигналами прекращения выборки являются или переполнение очереди (количество команд настраиваемое), или использование общих интерфейсов командами и данными (приоритет за данными). Для архитектуры AMCS-96 экспериментально была выбрана очередь команд в 10 инструкций, независимо от их длины. Механизм выполнения команд загружает готовую инструкцию в декодер микроопераций, формирующий сигналы внутренних фаз инструкций. Конец выполнения инструкции задается через специальную микрооперацию. Такая независимость механизмов позволяет вводить в функционал ядра команды, которые не имеют ограничений ни по длине кода, ни по времени выполнения.
- Конфигурируемые через параметры конфигурационного файла разрядности всех шин ядра. При создании модели ядра сразу была предусмотрена возможность увеличения разрядности шины данных, шины команд, шины адресов.
- Минимизация задержек в каждом из механизмов выполнения или выборки инструкций. Максимальное распараллеливание внутренних процессов при выполнении команд, возможность выполнения нескольких несвязанных микроопераций одновременно.
- Использование современных возможностей кремниевых технологий и САПР: применение однотактных умножителей, делителей, сдвигателей и т. д.

Для изделий 1874ВЕ96Т и 1874ВЕ7Т, при всей их похожести на предыдущие разработки ОАО «НИИЭТ», выполненные на оригинальном ядре MCS-96, данный подход позволил увеличить производительность. По итогам исследования по тестам CodeMark, выполненного фирмой «Ангиоскан» — нашим партнером по созданию ИСР, производительность увеличилась в 3–4 раза при функционировании на одной и той же частоте и при использовании одинаковых режимов работы с внешней памятью команд, что и оригинальные контроллеры MCS-96.

«Обработка-28»

В 2015 году ОАО «НИИЭТ» приступило к выполнению ОКР «Обработка-28», в рамках которой по требованиям пользователей предполагается модернизация радиационностойкого МК 1874ВЕ7Т, в том числе и в вопросах вычислительных мощностей и возможностей ядра. Основные недостатки процессорной части 1874ВЕ7Т:

1. Малый размер адресного пространства. Для обеспечения совместимости с МК, базирующимися на оригинальных MCS-96-архитектурах, ИМС 1874ВЕ7Т имеет такую же, как и они, адресную область 64 кбайт (16-разрядная шина адреса). Минимально требуемый размер на сегодня составляет 16 Мбайт (24-разрядная шина адреса).
2. Отсутствие внутренних областей памяти, из которых возможно выполнение программ. Использование этой области позволило бы обойти ошибки, возникающие во внешней памяти при обнаружении их кодами коррекции.

Помимо реализации пользовательских предложений, было решено модернизировать ядро следующим образом:

1. Повысить производительность путем перехода на 32-разрядную шину данных, добавления 32-битного АЛУ и нового набора команд.
2. Ввести подсистему арифметических команд, выполняемых за один такт. Данная подсистема получила название RISC-подсистемы, так как однотактовые команды работают с ограниченным количеством регистров.
3. Увеличить разрядность основного для радиационно-стойкого микроконтроллера интерфейса обращений к внешней памяти до 32 бит.
4. Добавить поддержку операций с плавающей точкой IEEE-754 с одинарной (32-бит) и двойной (64-бит) точностью.
5. Расширить адресное пространство до 4 Гбайт (32-разрядная шина адреса).
6. Ввести возможности гибкого управления параметрами доступа к внешней памяти с функциями помехоустойчивого кодирования для исправления ошибок доступа.
7. Увеличить количество векторов прерываний и ускорить запуск обработчиков прерываний.
8. Ввести механизмы защиты стека от переполнения/опустошения.

Поддержка однотактных команд и увеличенного адресного пространства вводится как для системы команд MCS-96, так и для системы новых 32-битных инструкций. Ранее созданная CISC-платформа позволила сократить время, требуемое для создания ядра ОКР «Обработка-28». В настоящее время все вышеописанные функции уже реализованы в Verilog-коде, и начинается активный процесс его тестирования. Далее будут подробно описаны особенности нового 32-разрядного ядра.

Подсистема выполнения команд

Процессорное ядро, используемое в ОКР «Обработка-28», имеет два режима функционирования ее системы команд:

1. Режим совместимости с MCS-96. В данном режиме микроконтроллер выполняет инструкции системы команд MCS-96. Единственное отличие в том, что часть не используемых в MCS-96 кодов инструкций в этом режиме применяется для новой функциональности. Например, в системе команд MCS-96 нет механизма переходов по адресам свыше 64 кбайт. Для этого в новом ядре во всех режимах работы присутствуют инструкции «очень длинных» переходов и вызовов процедур (VLCALL, VLJMP), которые позволяют осуществить переход на любой адрес адресного пространства. Для поддержки нового размера адресного пространства внесены изменения и в механизмы выполнения некоторых инструкций MCS-96. Теперь при вызове прерываний или процедур сохраняется 32-разрядный адрес точки возврата.
2. Режим выполнения 32-разрядных инструкций. В данном режиме вместо части MCS-96-инструкций становятся доступны инструкции работы с 32-разрядными данными. Так, появляются новые инструкции загрузки, арифметические, логические инструкции, инструкции работы со стеком. В этом режиме доступны и команды работы с байтами системы команд MCS-96.

В таблицах 1 и 2 представлена система команд микроконтроллера в первом и втором режимах работы ядра соответственно.

Переход из одного режима работы ядра в другой осуществляется инструкцией SWC. В систему команд введена еще одна специальная инструкция — команда конца инициализации EINIT. После первого запуска этой команды становится недоступной для модификации часть системных регистров микроконтроллера — регистры настроек доступа к внешней шине, регистры определения границ стека, регистры определения начальных адресов подпрограмм обработчиков прерываний.

Максимальная длина команд во всех режимах работы ядра осталась раной 7 байт, но благодаря масштабируемости Verilog-описания возможно введение инструкций и большей длины.

Увеличение разрядности шины данных и появление 32-разрядного АЛУ — это только одно из наиболее очевидных решений, позволяющих увеличить производительность. Вторым направлением стало введение команд, выполняемых за один такт. Поскольку сделать это для всего регистрового файла, к которому в архитектуре MCS-96 относится ОЗУ, технически невоз-

Таблица 1. Система команд в режиме MCS-96

Код	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	SKIP	CLR	NOT di	NEG di	XCH di	DEC di	EXT	INC di	SHR SHRB	SHL SHLB	SHRA SHRAB	XCH in XCHB	SHRL	SHLL	SHRAL	NORML
1x	SWC	CLRB	NOTB	NEGB	XCHB di	DECB	EXTB	INCB				in	***	***	***	***
2x	SJMP								SCALL							
3x	JBC								JBS							
	bit 0	bit 1	bit 2	bit 3	bit 4	bit 5	bit 6	bit 7	bit 0	bit 1	bit 2	bit 3	bit 4	bit 5	bit 6	bit 7
4x	AND 3op				ADD 3op				SUB 3op				MULU 3op *			
	di	im	in	ix	di	im	in	ix	di	im	in	ix	di	im	in	ix
5x	ANDB 3op				ADDB 3op				SUBB 3op				MULUB 3op *			
	di	im	in	ix	di	im	in	ix	di	im	in	ix	di	im	in	ix
6x	AND 2op				ADD 2op				SUB 2op				MULU 2op *			
	di	im	in	ix	di	im	in	ix	di	im	in	ix	di	im	in	ix
7x	ANDB 2op				ADDB 2op				SUBB 2op				MULUB 2op *			
	di	im	in	ix	di	im	in	ix	di	im	in	ix	di	im	in	ix
8x	OR				XOR				CMP				DIVU *			
	di	im	in	ix	di	im	in	ix	di	im	in	ix	di	im	in	ix
9x	ORB				XORB				CMPB				DIVUB *			
	di	im	in	ix	di	im	in	ix	di	im	in	ix	di	im	in	ix
Ax	LD				ADDC				SUBC				LDBZE			
	di	im	in	ix	di	im	in	ix	di	im	in	ix	di	im	in	ix
Bx	LDB				ADDCB				SUBCB				LDBSE			
	di	im	in	ix	di	im	in	ix	di	im	in	ix	di	im	in	ix
Cx	ST	BMOV	ST		STB	CMPL	STB		PUSH				POP	BMOVI	POP	
	di	–	in	ix	di	di	in	ix	di	im	in	ix	di	–	in	ix
Dx	JNST	JNH	JGT	JNC	JNVT	JNV	JGE	JNE	JST	JH	JLE	JC	JVT	JV	JLT	JE
Ex	DJNZ	DJNZW	TIJMP	BR in	VLCALL	F-инстр.	VLJMP	LJMP	***	***	***	***	DPTS	EPTS	прерыв.	LCALL
Fx	RET	TRAP1	PUSHF	POPF	PUSHA	POPA	MOD	TRAP	CLRC	SETC	DI	EI	CLRVT	NOP	*	RST

Обозначение	Прямая		Непосредственная		Косвенная (1)		Индексная S/L (1)	
DIV	4	FE 8C	5	FE 8D	4	FE 8E	5/6	FE 8F
DIVB	4	FE 9C	4	FE 9D	4	FE 9E	5/6	FE 9F
MUL (2 ops)	4	FE 6C	5	FE 6D	4	FE 6E	5/6	FE 6F
MUL (3 ops)	5	FE 4C	6	FE 4D	5	FE 4E	6/7	FE 4F
MULB (2 ops)	4	FE 7C	4	FE 7D	4	FE 7E	5/6	FE 7F
MULB (3 ops)	5	FE 5C	5	FE 5D	5	FE 5E	6/7	FE 5F

Таблица 2. Система команд в режиме выполнения инструкций работы с 32-битными данными (оранжевым цветом выделены новые инструкции)

Код	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	SKIP	ECLR	ENOT di	ENEG di	ELDS di	EDEC di	EXT	EINC di	SHR SHRB	SHL SHLB	SHRA SHRAB	XCH in XCHB	ESHRL di	ESHLL di	ESHRAL di	NORML
1x	SWC	CLRB	NOTB	NEGB	XCHB di	DECB	EXTB	INCB				in	EST shot ix pc			
2x	SJMP								SCALL							
3x	JBC								JBS							
	bit 0	bit 1	bit 2	bit 3	bit 4	bit 5	bit 6	bit 7	bit 0	bit 1	bit 2	bit 3	bit 4	bit 5	bit 6	bit 7
4x	AND 3op				ADD 3op				SUB 3op				MULU 3op *			
	di	im	in	ix	di	im	in	ix	di	im	in	ix	di	im	in	ix
5x	ANDB 3op				ADDB 3op				SUBB 3op				MULUB 3op *			
	di	im	in	ix	di	im	in	ix	di	im	in	ix	di	im	in	ix
6x	EAND 2op				EADD 2op				ESUB 2op				EMULU 2op *			
	di	im	in	ix	di	im	in	ix	di	im	in	ix	di	im	in	ix
7x	ANDB 2op				ADDB 2op				SUBB 2op				MULUB 2op *			
	di	im	in	ix	di	im	in	ix	di	im	in	ix	di	im	in	ix
8x	EOR				EXOR				CMP				DIVU *			
	di	im	in	ix	di	im	in	ix	di	im	in	ix	di	im	in	ix
9x	ORB				XORB				CMPB				DIVUB *			
	di	im	in	ix	di	im	in	ix	di	im	in	ix	di	im	in	ix
Ax	ELD				EADDC				ESUBC				ELD long			
	di	im	in	ix	di	im	in	ix	di	im	in	ix	ix pc	ix pc	ix pc	ix pc
Bx	LDB				ADDCB				SUBCB				ELD shot			
	di	im	in	ix	di	im	in	ix	di	im	in	ix	ix pc	ix pc	ix pc	ix pc
Cx	ST	BMOV	EST		STB	ECMPL	STB		PUSH				EPOP	BMOVI	EPOP	
	di	–	in	ix	di	di	in	ix	di	im	in	ix	di	–	in	ix
Dx	JNST	JNH	JGT	JNC	JNVT	JNV	JGE	JNE	JST	JH	JLE	JC	JVT	JV	JLT	JE
Ex	DJNZ	DJNZW	TIJMP	BR in	VLCALL	F-инстр.	VLJMP	LJMP	EST long ix pc				DPTS	EPTS	прерыв.	LCALL
Fx	RET	TRAP1	PUSHF	POPF	PUSHA	POPA	MOD	TRAP	CLRC	SETC	DI	EI	CLRVT	NOP	*	RST

Обозначение	Прямая		Непосредственная		Косвенная (1)		Индексная S/L (1)	
DIV	4	FE 8C	5	FE 8D	4	FE 8E	5/6	FE 8F
DIVB	4	FE 9C	4	FE 9D	4	FE 9E	5/6	FE 9F
MUL (2 ops)	4	FE 6C	5	FE 6D	4	FE 6E	5/6	FE 6F
MUL (3 ops)	5	FE 4C	6	FE 4D	5	FE 4E	6/7	FE 4F
MULB (2 ops)	4	FE 7C	4	FE 7D	4	FE 7E	5/6	FE 7F
MULB (3 ops)	5	FE 5C	5	FE 5D	5	FE 5E	6/7	FE 5F

можно, была определена область адресного пространства, для которого однотактные команды будут доступны. Таким образом, первые адреса адресного пространства, начиная с адреса 0000_0004h и заканчивая адресом 0000_0044h, образуют область 32-разрядных регистров R0-R15, при попадании операндов в которую используется свой набор внутренних микроопераций, что позволяет существенно ускорить выполнение команд. Так как определить, попадают ли операнды в область R0-R15, нужно в тот же такт, когда команда загружается в декодер микроопераций, подобное ускорение можно сделать только для команд, использующих прямую адресацию, поскольку лишь при таком виде адресации конечные адреса операндов определены в самой команде. Если при прямой адресации операнды и результат попадают в область R0-R15, то выполнение такой команды происходит с помощью специальной подсистемы (назовем ее RICS, так как сокращен набор адресов, с которой она работает) за один такт. В таблицах 1 и 2 команды, имеющие свой RISC-эквивалент выполнения, выделены синим цветом. Как видно на рисунках, RISC-подсистема действует во всех режимах работы ядра. Например, команда ADD 20h,22h из системы команд MCS-96 будет выполняться за один такт, а команда ADD 60h,22h — за четыре. Особый случай представляют собой команды с тремя операндами и командами сдвига. Для первых существует два типа ускорения: в первом случае, если все три операнда попадают в область регистров, выполнение происходит за один такт; в случае если в область регистров попадают только входные операнды, а результат необходимо поместить в область ОЗУ, то выполнение команды происходит за два такта. Для команд сдвига, помимо условия нахождения операнда в области R0–R15, существует второе условие ускорения — количество сдвигов должно быть указано в самой команде.

Механизм RISC-подсистемы предоставляет дополнительные возможности по оптимизации кода по объему/скорости выполнения, поскольку для одних случаев позволяет минимизировать количество команд (для выполнения арифметических действий не требуется пересылка из области ОЗУ в область регистров R0–R15 дополнительными командами), а для других — минимизировать время выполнения. Архитектуру микроконтроллера можно назвать CISC+RISC-архитектурой, достоинством которой является то, что CISC- и RISC-команды имеют одинаковый код.

Еще один дополнительный блок ядра микроконтроллера, позволяющий увеличить производительность, но уже в операциях с плавающей запятой,— это блок FPU, реализующий стандарт IEEE-754. Для выполнения новых команд были введены так называемые F-инструкции. Блок FPU работает со своим набором 32-разрядных регистров FR0–FR1, адресация к которым доступна как специальными инструкциями FLD, FST по коротким адресам, так и по адресам общего адресного пространства. Блок FPU поддерживает операции с одинарной (32-бит) и с двойной (64-бит) точностью. Вычисление результата блоком FPU возможно каждый такт.

Подсистема выборки команд

Для обеспечения максимальной производительности необходимо иметь такую подсистему выборки команд, чтобы она как можно быстрее подготавливала инструкцию к исполнению. В условиях, когда длина команд нефиксированная, повысить производительность только увеличением разрядности интерфейсов доступа к командам нельзя. Например, выбрав 64 бит из области команд (при длине команд от 8 до 56 бит), за один такт определить все находящиеся в этом блоке команды технически невозможно (конечно, разумными для этой задачи ресурсами). Но, с другой стороны, для некоторого набора команд при определенных условиях (именно условиях, а не требованиях) увеличение разрядности интерфейса позволяет ускорить выполнение программ. Исходя из этого, подсистема выборки команд в ИМС, создаваемой в ходе ОКР «Обработка-28», получила ряд улучшений относительно других контроллеров архитектуры MCS-96.

1. Появление так называемой истории команд на 32 (в текущей версии) инструкции. Если осуществляется переход в рамках последних выполняемых 32 инструкций, то блок выборки команд не обращается заново к внешней памяти или внутренней PSRAM, а загружает очередь команд из сохраненной ранее области. Данный механизм отключаемый. Размер области истории команд был выбран таким первоначально и является обсуждаемым с потенциальными потребителями.
2. Увеличена очередь команд. Размер ее составляет 16 инструкций.
3. Введен новый 32-разрядный режим доступа к внешней памяти команд. Его особенности будут описаны ниже.
4. Введен механизм обращения к различным областям внешней памяти с индивидуальными настройками параметров доступа. Микроконтроллер имеет набор из восьми пар 32-разрядных регистров выбора границ окон (ADDRSELx_LO, ADDRSELx_HI) и восьми управляющих параметрами доступа регистров (BUSCONx). Для каждой из областей возможны индивидуальные настройки следующих параметров: количество тактов ожидания при обращении к шине, активный уровень сигнала CS (chip select), разрешение сигнала CS, использование помехоустойчивого кодирования, тип доступа (8-, 16-, 32-разрядный).

По сравнению с архитектурой MCS-96 микроконтроллер «Обработка-28» имеет еще один режим доступа к внешней шине — 32-разрядный. В данном режиме подсистема выборки команд получает сразу 4 байт информации, и начинается процесс их декодирования. Длительность такого процесса составляет от одного до четырех тактов. В случае если команды выравнены по адресам, кратным 4, и в блоке информации присутствует только одна команда (4-байт), то декодирование и размещение такой инструкции в очереди команд будет самым быстрым. Самое долгое декодирование происходит, когда из внешней памяти выбраны четыре однобайтные инструкции. В этом случае их размещение в очереди команд займет четыре такта. Механизм работы подсистемы выборки аналогичен процессу распаковки информации: чем больше в полученном блоке информации (в данном случае инструкций), тем дольше он идет. Этот процесс автоматический, и он наглядно представлен на рис. 1 — длительность внешнего сигнала RD пропорциональна времени размещения команд в очереди. Возможность ускорения выборки из памяти дает еще одно направление в процессе оптимизации пользовательского кода.

В отличие от оригинальных микроконтроллеров архитектуры MCS-96, ИМС «Обработка-28» имеет область памяти типа ОЗУ (PSRAM), из которой возможно выполнение команд. В данной области предусмотрено хранение обработчиков прерываний, вызванных ошибками в командах или

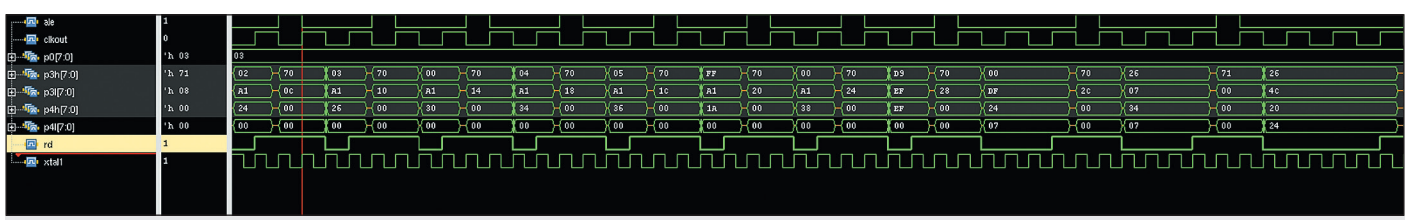

Рис. 1. Обращение к внешней памяти по 32-разрядному интерфейсу

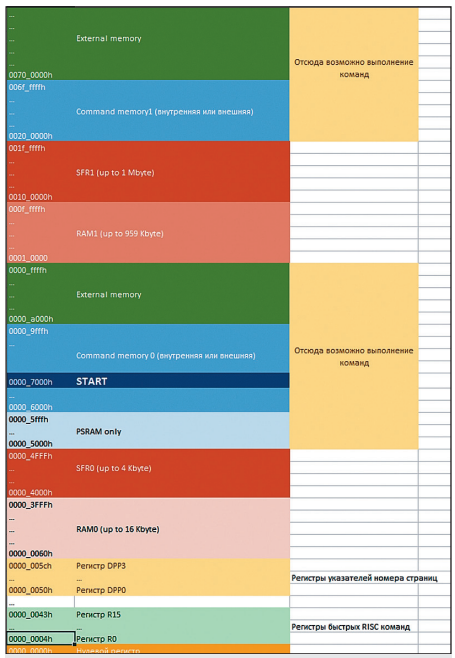

Рис. 2. Адресное пространство микроконтроллера

данных, размещенных во внешней памяти в случае использования помехоустойчивого кодирования. Доступ к этому виду памяти осуществляется по 32-разрядной шине, и его скорость так же зависит от набора команд, как и для режима внешней памяти.

Адресное пространство

Микроконтроллер «Обработка-28» имеет 4 Гбайт адресного пространства. Карта памяти представлена на рис. 2.

Старт микроконтроллера происходит с адреса 0000_7000h.

Система команд MCS-96 не предусматривает команд переходов и способов адресации для области памяти свыше 64 кбайт. Для поддержки функциональности новой адресной области были проведены следующие мероприятия:

1. Для осуществления переходов и запусков подпрограмм в обоих режимах работы ядра появились команды «очень длинных» переходов VLCALL, VLJMP. Также при вызовах подпрограмм инструкциями SCALL, LCALL при переходах на подпрограммы обработчиков прерываний в стеке сохраняется полный 32-разрядный адрес.
2. Для доступа к данным вводится механизм адресации с помощью указателей страниц данных (data page pointers — DPP).

Микроконтроллер имеет четыре указателя страниц данных. Они позволяют расширить адрес в любом из режимов функционирования ядра до 32 разрядов. В случае работы ядра в режиме MCS-96 действуют только два указателя из четырех. Формирование адреса в этом случае происходит, как показано на рис. 3.

Особенность использования указателей состоит в том, что они расширяют адрес только при микрооперациях прямой адресации. Какой из указателей DPP будет применен при формировании адреса, решает старший бит адреса из команды. Рассмотрим это на примере. В случае выполнения команды ADD 60h, 62h запускаемые микрооперации будут выглядеть следующим образом:

1. Происходит чтение по прямому адресу значения ячейки 62h. Значение попадает в АЛУ.
2. Происходит чтение по прямому адресу значения ячейки 60h. Значение попадает в АЛУ.
3. Результат операции после вычисления по прямому адресу заносится в ячейку 60h.

В данном случае во всех трех стадиях выполнения команды используется прямая адресация, соответственно, во всех них будет использоваться DPP. В случае выполнения команды ADD 60h, [62h] запускаемые микрооперации будут выглядеть следующим образом:

1. Происходит чтение по прямому адресу значения ячейки 62h.
2. Происходит чтение с использованием косвенной адресации по адресу, взятому из ячейки 62h. Это значение попадает в АЛУ.
3. Происходит чтение по прямому адресу значения ячейки 60h. Значение попадает в АЛУ.
4. Результат операции после вычисления по прямому адресу заносится в ячейку 60h.

В данном случае DPP используется при адресации всегда, кроме шага 2. Во всех ситуациях, когда адрес взят из ячейки памяти, указатели страниц не используются. Особенностью адресации в данном режиме работы ядра является то, что косвенной адресации доступно только 64 кбайт, как и в MCS-96. То есть ячейка с адресом может находиться в любом месте адресного пространства, но оттуда считается только 16 бит конечного адреса.

Для режима работы с 32-разрядными данными применяются все четыре указателя страниц. Формирование адреса при этом представлено на рис. 4.

Особенности работы в данном режиме: определяется, какой будет использоваться указатель, предусмотрено уже два бита адреса команды; при операции чтения или записи по адресу, заданному косвенно, из ячейки памяти считывается 32 бит адреса. То есть и ячейка с указателем, и сама ячейка, откуда берутся данные для операции, могут находиться в любом месте адресного пространства.

Указатели на страницу данных не используются для адресов, участвующих в прямой адресации и меньших 0000_0060h. В этой области находятся регистры R0–R15 и сами указатели DPP.

Работа с прерываниями

Нововведения в работе с прерываниями в ИМС «Обработка-28» связаны со следующими особенностями схемы:

1. Увеличение числа периферийных блоков и рост их функциональности привели к необходимости повысить количество векторов прерывания. В данной редакции ядра их используется 64 штуки. Управляются прерывания посредством двух 32-разрядных регистров INT_MASK и INT_MASK2. Запросы на прерывания ожидают обработки в регистрах INT_PEND и INT_PEND2.
2. Появление в архитектуре новых служебных прерываний: прерывания по ошибкам внешнего интерфейса, прерывания по переполнению и опустошению стека, прерывания по ошибкам доступа к регистрам FPU.
3. Изменение области расположения векторов прерывания. Связано с появлением помехоустойчивого кодирования интерфейса внешней памяти. В архитектуре MCS-96 векторы располагаются в той же области, что и команды. Если действует режим работы с внешней памятью, то и векторы будут находиться в ней. Это могло бы привести к парадоксальной ситуации: для того чтобы правильно выйти

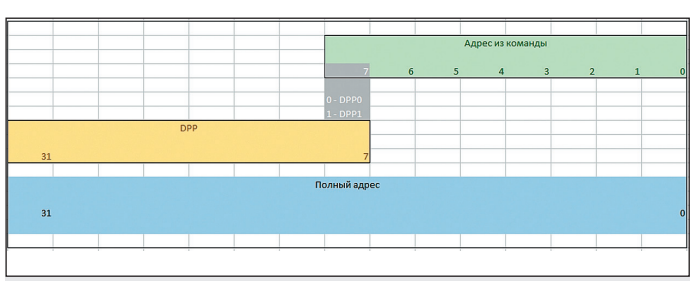

Рис. 3. Формирование адреса операнда в случае работы ядра в режиме MCS-96

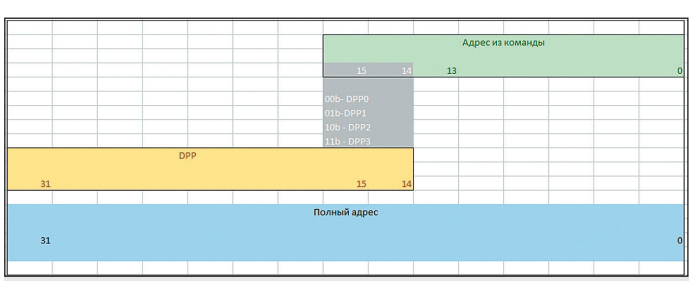

Рис. 4. Формирование адреса в режиме работы с 32-разрядными данными

на обработчик прерывания по ошибке внешнего интерфейса, микроконтроллер должен считать адрес перехода из той же внешней памяти. В ИМС «Обработка-28» адреса обработчиков прерываний хранятся в регистрах (INTx_ADDR), находящихся в области SFR. По умолчанию адреса настроены на расположение обработчиков во внешней памяти, но могут быть и переназначены на область PSRAM (которую тоже предполагается защитить от сбоев). Регистры INTx_ADDR должны быть установлены в инициализационной части программы пользователя и не могут быть изменены после выполнения команды EINIT. Подход с расположением векторов в области SFR также позволил ускорить переход на обработчик прерывания, поскольку ликвидировано одно (с точки зрения пользовательской программы не несущее полезной функциональности) обращение к внешней памяти: после обработки запроса микроконтроллер сразу выходит на нужный адрес, в котором содержатся команды.

В микроконтроллере «Обработка-28» произошли изменения и в механизме обработки прерываний блоком периферийных транзакций (PTS). Из всего многообразия режимов работы PTS архитектуры MCS-96 в новом микроконтроллере остались только режимы одиночной и блочной передачи. Эти режимы являются универсальными и позволяют вести пересылку данных между любыми ячейками всего адресного пространства. Еще одно нововведение в работе блока PTS — возможность выбора для каждого сеанса обмена типа приоритета: приоритет ядра или приоритет PTS. В первом случае блок периферийных транзакций отслеживает моменты, когда не используется внутренняя память (например, запущена микрооперация вычисления в АЛУ или простой шины вследствие неготовности команды для загрузки в декодер микроопераций), и в эти моменты ведет свой обмен; во втором — PTS остановит выполнение команд ядром и выполнит обмен данными максимально быстро.