Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования

Новгородский государственный университет
имени Ярослава Мудрого

Разработка программного обеспечения для сигнальных процессоров TMS320C64xx

Новгород
2011

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
Новгородский государственный университет
имени Ярослава Мудрого
Институт электронных и информационных систем

Кафедра радиосистем

Вахлачёв Н.В.

Разработка программного обеспечения для сигнальных процессоров TMS320C64xx

Методическое пособие по дисциплине
"СИГНАЛЬНЫЕ ПРОЦЕССОРЫ"

для специальности 210302.65 "Радиотехника"

Новгород
2011

УДК 621.38
ББК

Вахлачев Н.В. Разработка программного обеспечения для сигнальных процессоров TMS320C64xx: Методическое пособие / ФГБОУ «Новгородский государственный университет им. Ярослава Мудрого», Великий Новгород, 2011 г. - 112 с.

В учебном пособии рассмотрена архитектура сигнальных процессоров фирмы Texas Instruments семейства TMS320C64xx, периферийные устройства и интегрированная среда разработки ПО Code Composer Studio.

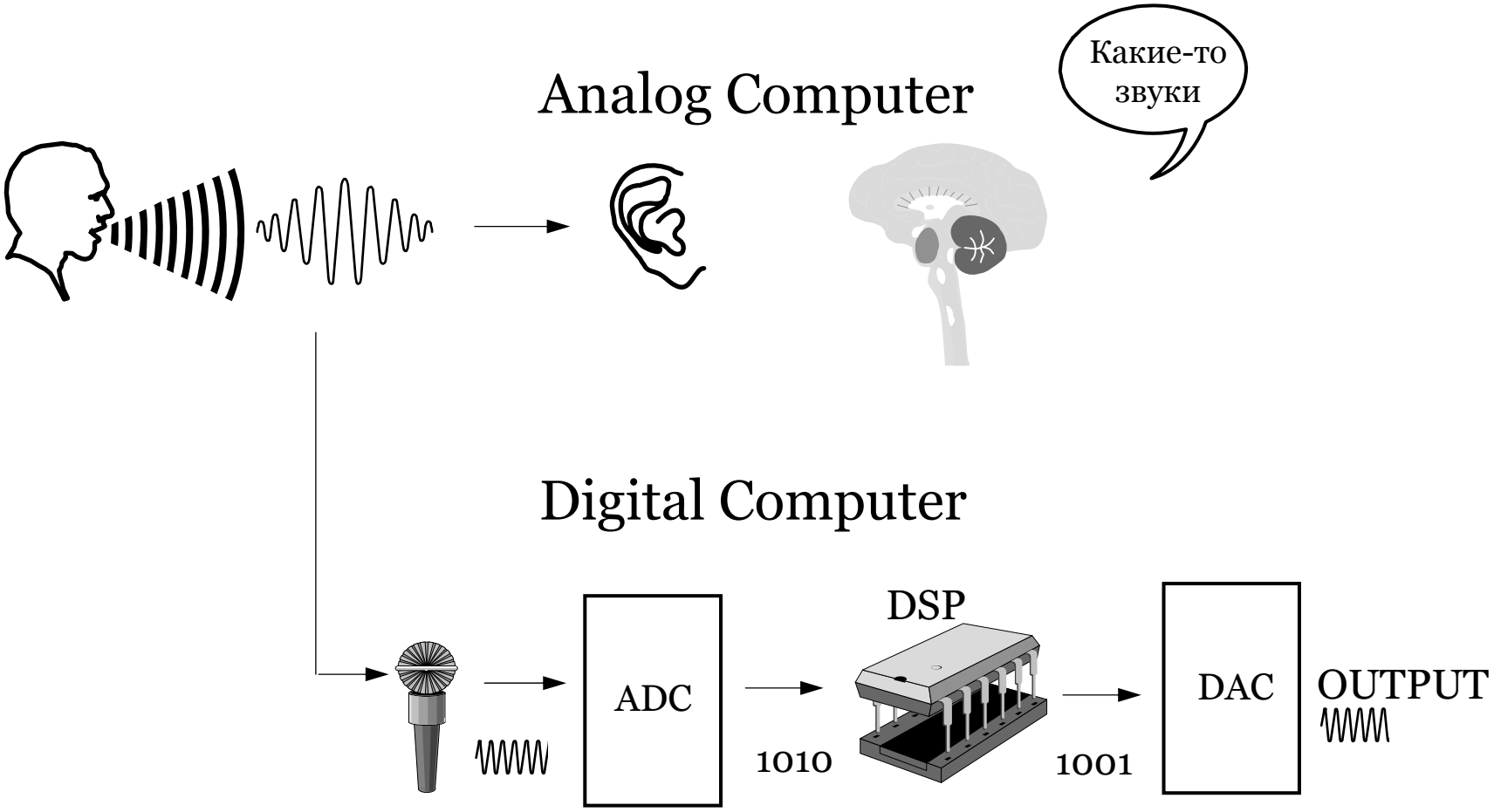
Учебное пособие отвечает новым образовательным стандартам и предназначено для подготовки специалистов по специальности 210302.65 "Радиотехника".

Учебное пособие одобрено советом института электронных и информационных систем Новгородского государственного университета имени Ярослава Мудрого.

Разработка программного обеспечения для сигнальных процессоров TMS320C64xx

Часть 1. Основные понятия.

Что такое DSP?



Что такое DSP

Цифровой сигнальный процессор (*Digital signal processor*, *DSP*) — специализированный микропроцессор, предназначенный для цифровой обработки сигналов в реальном масштабе времени.

Типичные операции ЦОС

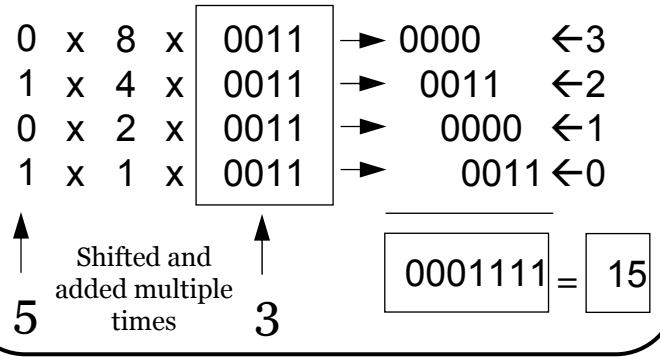
Add

1+2 = 3

```
  0001
+ 0010
------
  0011
```

Multiply

5*3 = 15



Наиболее часто используемая операция в ЦОС

$$A = B*C + D$$

$$E = F*G + A$$

$$\vdots$$

Multiply, Add, and Accumulate

MAC Instruction

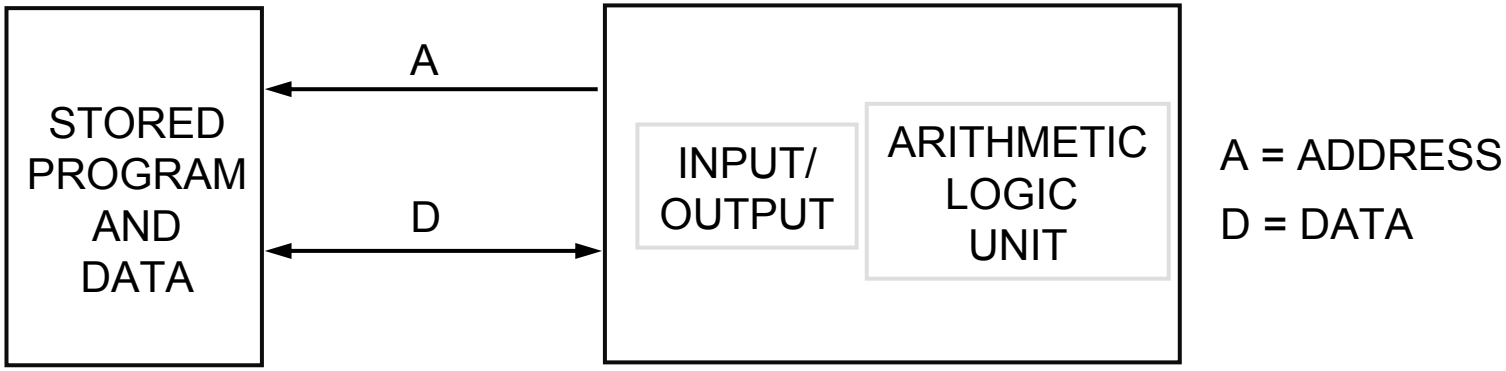
MAC Operation

Обычно 70 тактов для обычного процессора

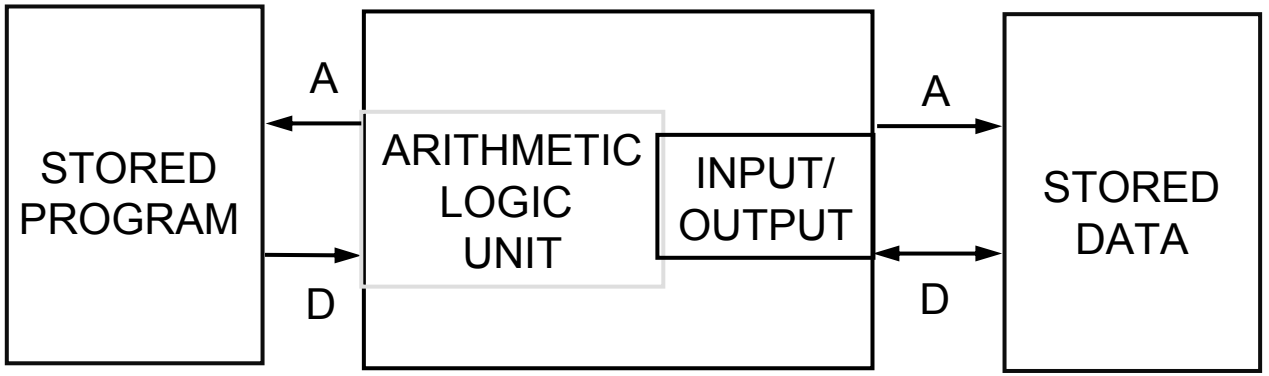
Обычно 1 такт для ЦСП

Два типа архитектуры процессор

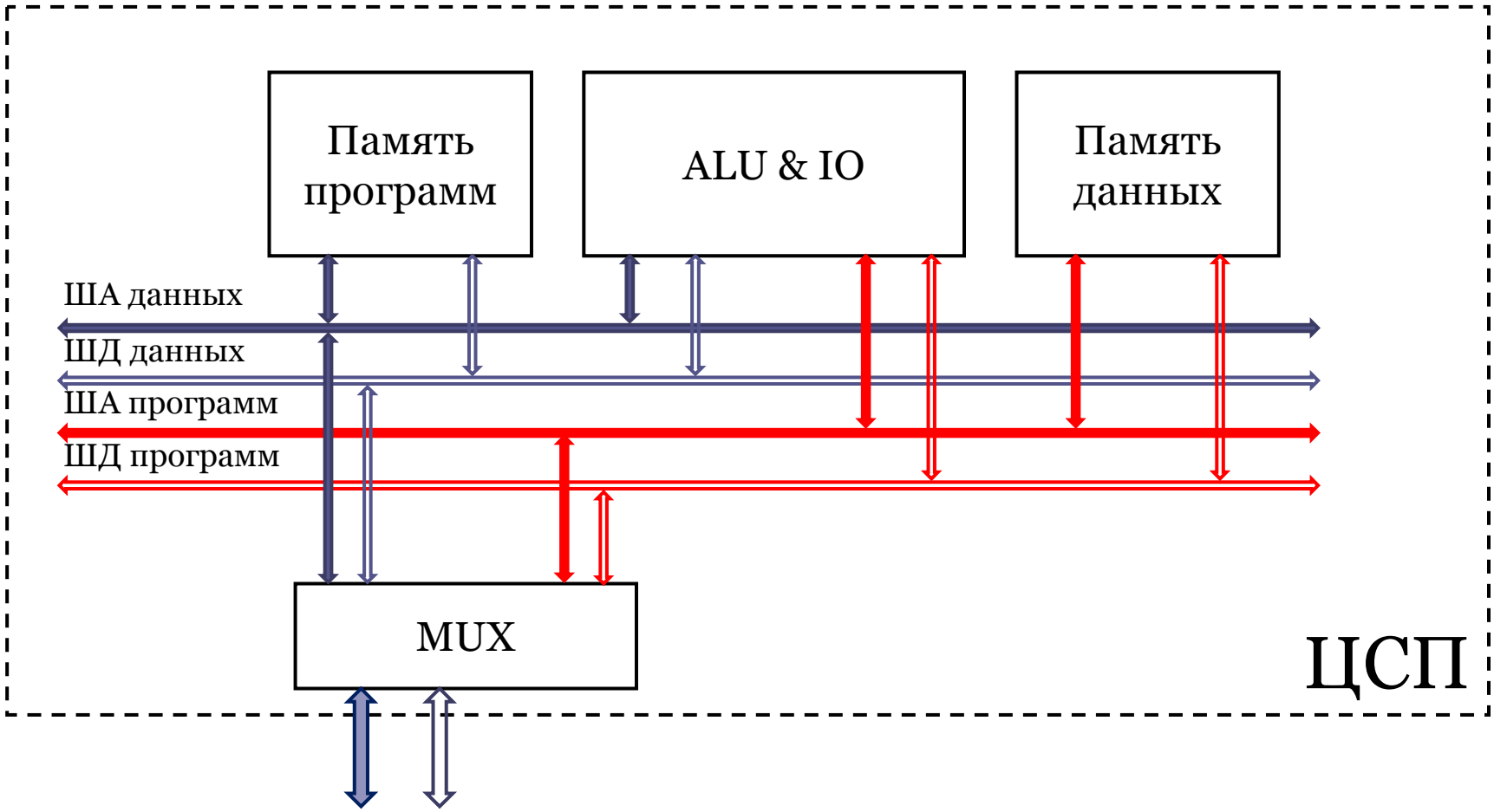
Архитектура Фон-Неймана



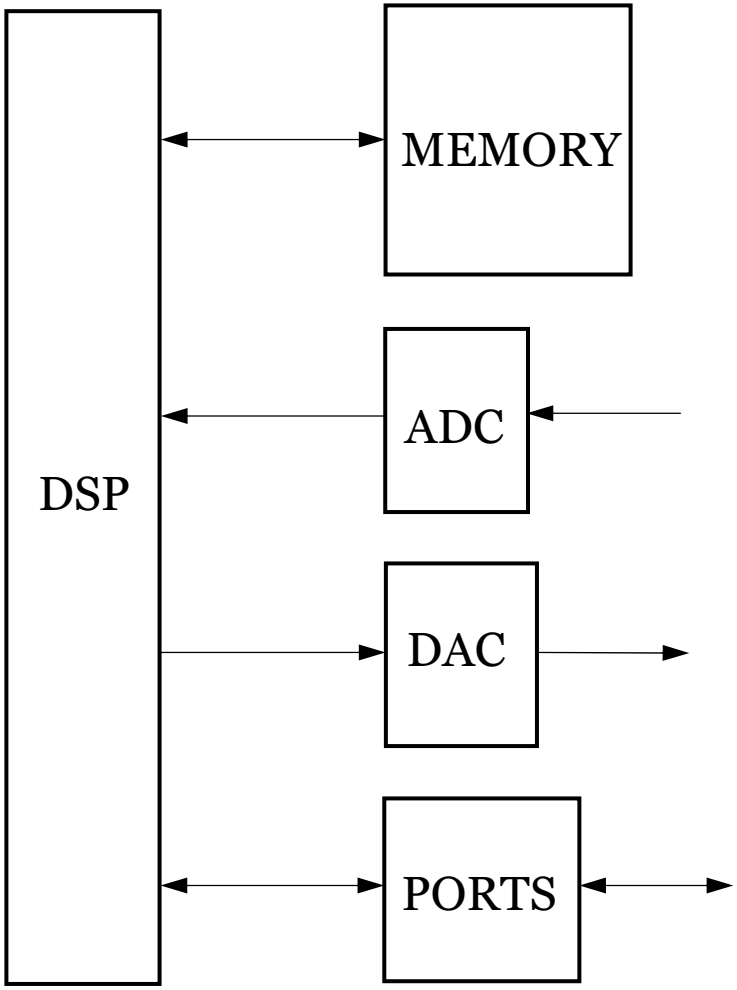
Гарвардская архитектура



Модифицированная Гарвардская архитектура



Типичная DSP система



- **DSP Chip**
- **Memory**
- **Converters (Optional)**
  - **Analog to Digital**
  - **Digital to Analog**
- **Communication Ports**
  - **Serial**
  - **Parallel**

Особенности архитектуры DSP

- Гарвардская архитектура (разделение памяти команд и данных);
- Наличие нескольких операционных модулей работающих параллельно;
- Наличие генераторов адресных последовательностей позволяющих достаточно просто организовать специфические виды адресации (бит-ревесную и циклическую);
- Наличие аппаратной организации циклов;
- Поддержка специальных арифметических операций характерных для ЦОС (например умножение с накоплением);
- Выполнение инструкций за один такт;
- Сравнительно небольшая длина конвейера инструкций.
- Наличие высокоскоростной встроенной ОЗУ.

Разработка программного обеспечения для сигнальных процессоров TMS320C64xx

Часть 2. Характеристики современных DSP.

Основные производители DSP и принадлежащие им доли рынка

Название компании	Доля рынка DSP
Texas Instruments	54,3%
Freescale Semiconductor	14,1%
Analog Devices	8,0%
Philips Semiconductors	7,5%
Agere Systems	7,3%
Toshiba	4,9%
DSP Group	2,2%
NEC Electronics	0,6%
Fujitsu	0,4%
Intersil	0,3%
Other Companies	0,5%

Области применения семейств DSP разных производителей

Область применения	Семейство/производитель
Обработка видео, видеонаблюдение, цифровые камеры, 3D графика	TMS320DM64x/DaVinci, TMS320C64xx, TMS320C62xx (TI), PNX1300, PNX1500, PNX1700 (Philips) , MPC52xx (Freescale)
Обработка аудио, распознавание речи, синтез звука	TMS320C62xx, TMS320C67xx (TI), SHARC (Analog Devices)
Портативные медиа устройства	TMS320C54xx, TMS320C55xx (TI), Blackfin (Analog Devices)
Беспроводная связь, телекоммуникации, модемы, сетевые устройства	TMS320C64xx, TMS320C54xx, TMS320C55xx (TI), MPC7xxx, MPC86xx, MPC8xx PowerQUICC I, MPC82xx PowerQUICC II, MPC83xx PowerQUICC II Pro, MPC85xx PowerQUICC III (Freescale), Blackfin, TigerSHARC (Analog Devices), PNX1300 (Philips)
Управление приводами, преобразование мощности, автомобильная электроника, предметы домашнего обихода, офисное оборудование	TMS320C28xx, TMS320C24xx (TI), ADSP-21xx (Analog Devices), MPC55xx, MPC55xx (Freescale)
Медицина, биометрия, измерительные системы	TMS320C62xx, TMS320C67xx, TMS320C55xx, TMS320C28xx (TI), TigerSHARC, SHARC (Analog Devices)

Формат обрабатываемых данных и разрядность DSP

- DSP использующие целочисленную арифметику (fixed-point):
  - 16 битные;
  - 32 битные;
- DSP использующие арифметику с плавающей точкой (floating-point):
  - 32 битные.

Системы счисления

- Десятичная (dec)
- Двоичная (bin)
  - $3 \rightarrow 0\cdot2^8 + 0\cdot2^7 + 0\cdot2^6 + 0\cdot2^5 + 0\cdot2^4 + 0\cdot2^3 + 0\cdot2^2 + 1\cdot2^1 + 1\cdot2^0 \rightarrow 00000011\ b$
- Шестнадцатеричная (hex)
  - $33 \rightarrow 0011\ 0011\ b \rightarrow 0x33$

Представление чисел в дополнительном коде

- Для кодирования используется дополнительный код, поскольку арифметические действия над числами со знаком, представленными в доп. коде, выполняются как над беззнаковыми.
- Порядок преобразования чисел в дополнительный код:
  - Для положительных чисел, дополнительный код совпадает с прямым;
  - Для отрицательных чисел все разряды числа инвертируются, а к результату прибавляется 1. К получившемуся числу дописывается старший (знаковый) разряд, равный 1.

Примеры работы с числами в дополнительном коде

- Преобразование в дополнительный код

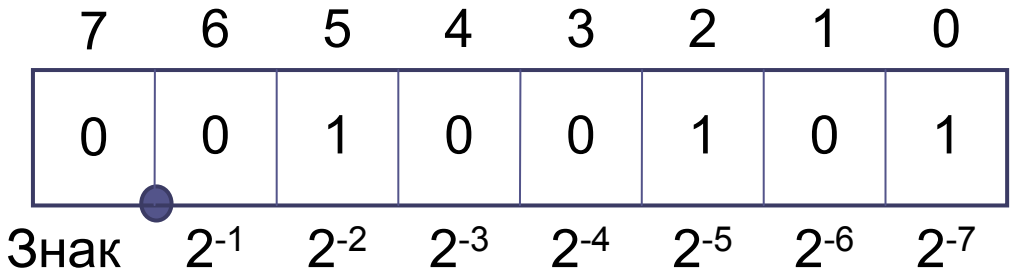
Действие	Dec	Bin							
		Знак	Число						
Исходное число	-2	1	000 0000 0000 0000 0000 0000 0000 0010						
Инверсия		1	111 1111 1111 1111 1111 1111 1111 1101						
Добавление 1		1	1						
Число в доп. коде		1	111 1111 1111 1111 1111 1111 1111 1110						
Hex		**F**	**F**	**F**	**F**	**F**	**F**	**F**	**E**

- Сложение чисел в дополнительном коде

Dec	Hex	Bin в дополнительном коде	
		Знак	Number
-2	FF FF FF FE	1	111 1111 1111 1111 1111 1111 1111 1110
3	00 00 00 03	0	000 0000 0000 0000 0000 0000 0000 0011
-2 + 3 = 1	00 00 00 01	0	000 0000 0000 0000 0000 0000 0000 0001

Целочисленная арифметика

- Использует представление дробных чисел в формате с фиксированной точкой (fixed-point notation).
  - Такое представление означает что в рамках заданного формата для всех чисел фиксируется одинаковое местоположение запятой, разделяющей знаковую и дробные части.
  - Например для числа 0,2890625 имеем:



  - $2^{-1}*0+2^{-2}*1+ 2^{-3}*0+ 2^{-4}*0+ 2^{-5}*1+ 2^{-6}*0+ 2^{-7}*1=$ 0,2890625

Q формат

- Q это формат чисел с фиксированной точкой, где указывается число бит дробной части и опционально количество бит целой части.
  - Запись Qm.n означает:
    - Q – означает, что число в Q формате;
    - m (опционально, обычно 0);
    - n – количество бит отводимых на представление дробной части в форме дополнения до двух.
- Для числа в формате Qm.n, требующего для хранения знаковое целое длиной m+n+1 справедливо следующее:
  - Диапазон значений $[-2^m, 2^m – 2^{-n}]$;
  - Разрешение 2^{-n}

Преобразования

- Из вещественного в целочисленное Qm.n:
  - Умножаем число с плавающей точкой на 2^n ;
  - Округляем к ближайшему целому;
- Из целочисленного Qm.n в вещественное:
  - Делим число в Q формате на 2^n

- Примеры:
  - 0.0098 -> Q31:

    $0.0098*2^{31}$ = 21045339.750 -> 21045340 -> 0x141205c;
  - 0xaf340001 (Q31) -> float:

    0xaf340001 -> -1355546623 /2^{31} = -0.6312255854718387;

Арифметика с плавающей точкой

TMS single-precision floating-point format

31	...	24	23	22	.....	0
e			**s**	**f**		
8 bits			1 bit	23 bits		

← Bit No

e = exponent is a signed two's compliment 8-bit field and determines the location of the binary Q point

s = sign of mantissa (s = 0 positive, s =1 negative)

f = fractional part of the mantissa; an implied 1.0 is added to this fraction but is not allocated in the bit field since this value is always present

Conversion equations

	Binary	**Decimal**	**Equation**
s = 0	$X = 01.f \times 2^e$	$X = 01.f \times 2^e$	1
s = 1	$X = 10.f \times 2^e$	$X = (-2 + 0.f) \times 2^e$	2

Special case

s = 0	X = 0	e = -128

Exponent (e)					
Decimal	0	1	127	-1	-128
Hex two's comp.	00	01	7F	FF	80

Fixed-point vs. Floating Point

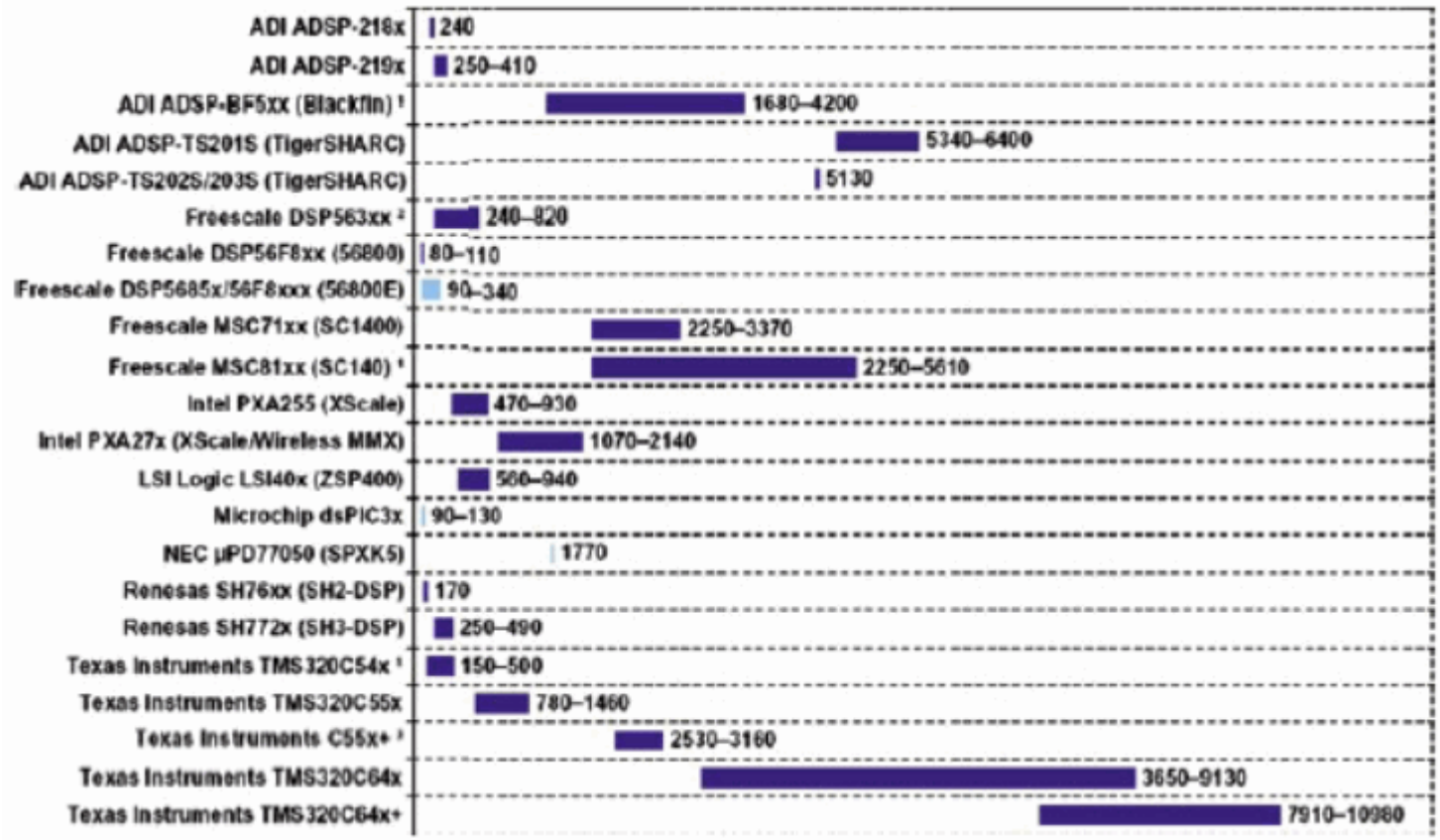
|DSP с фиксированной точкой

- быстрая и недорогая реализация;
- обеспечивают большую абсолютную точность при равной разрядности
- ограниченный диапазон представляемых чисел и как следствие подверженность проблемам переполнения;
- необходимость масштабирования результатов, что приводит к снижению достижимого отношения сигнал-шум;

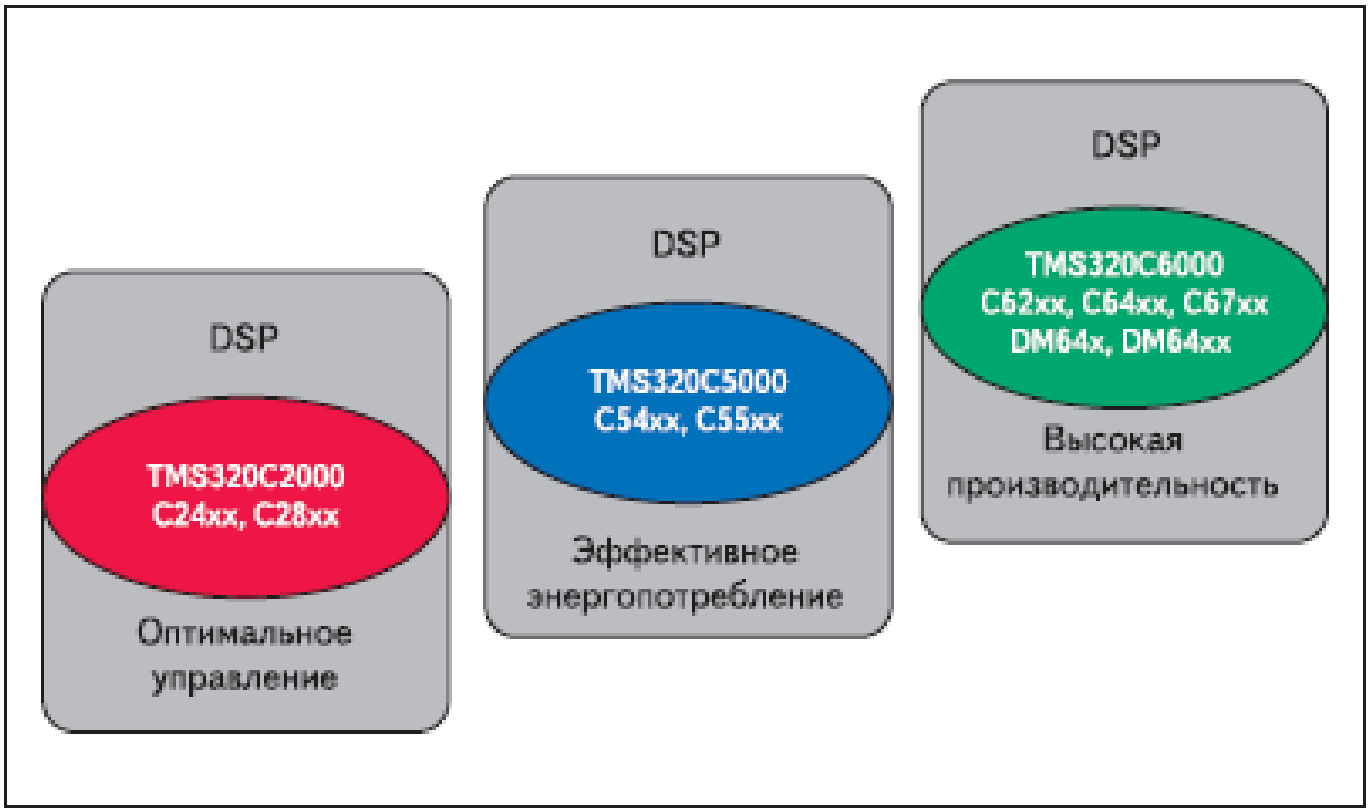
DSP с плавающей точкой

- Более дорогая и зачастую медленная реализация;
- Более широкий динамический диапазон, отсутствие проблемы переполнения
- Простота и естественность программирования;

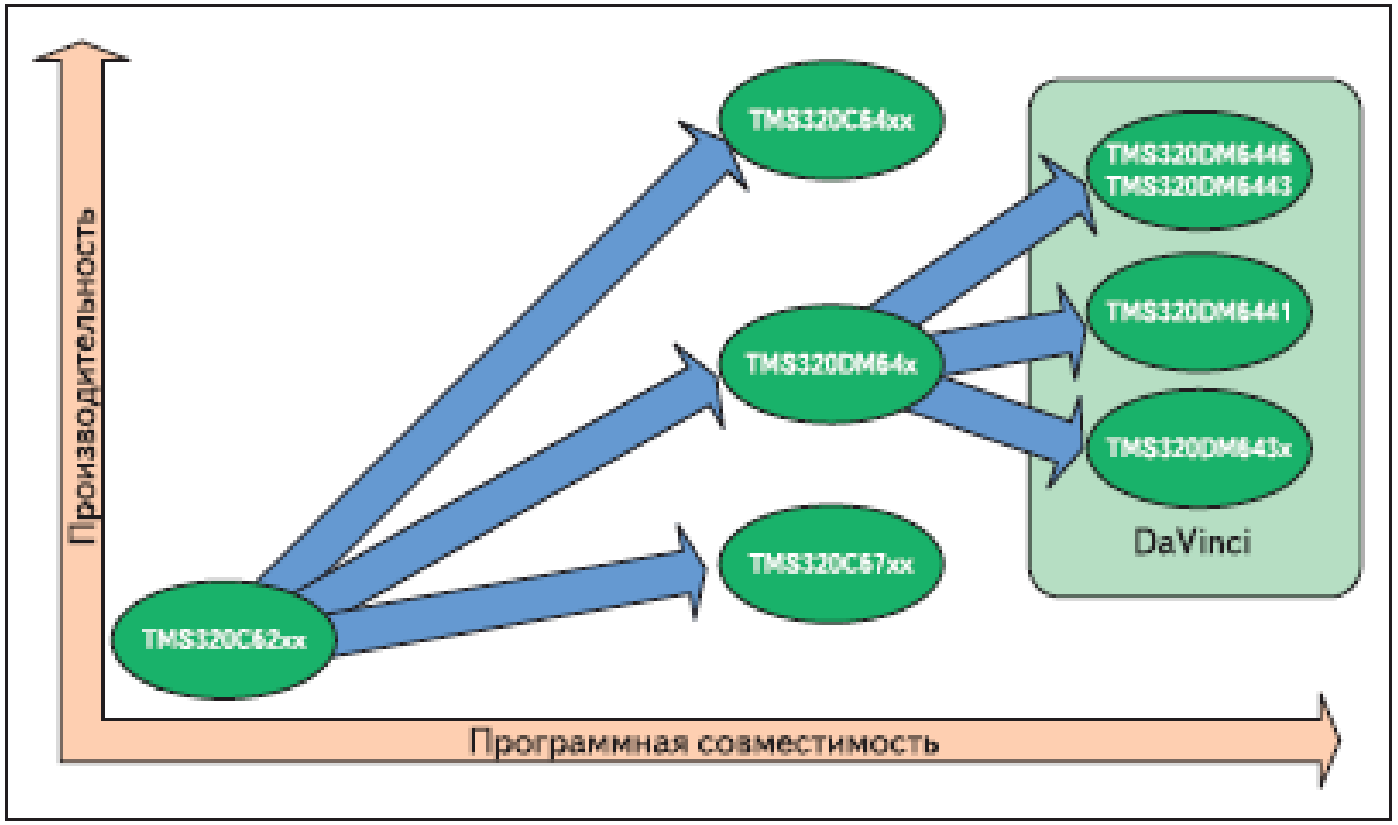
Сравнение быстродействия DSP с фиксированной точкой



DSP фирмы TI


DSP
TMS320C2000
C24xx, C28xx
Оптимальное
управление
DSP
TMS320C5000
C54xx, C55xx
Эффективное
энергопотребление
DSP
TMS320C6000
C62xx, C64xx, C67xx
DM64x, DM64xx
Высокая
производительность

Основные поколения DSP семейства TMS320C6000



Обобщенная структура DSP DaVinci

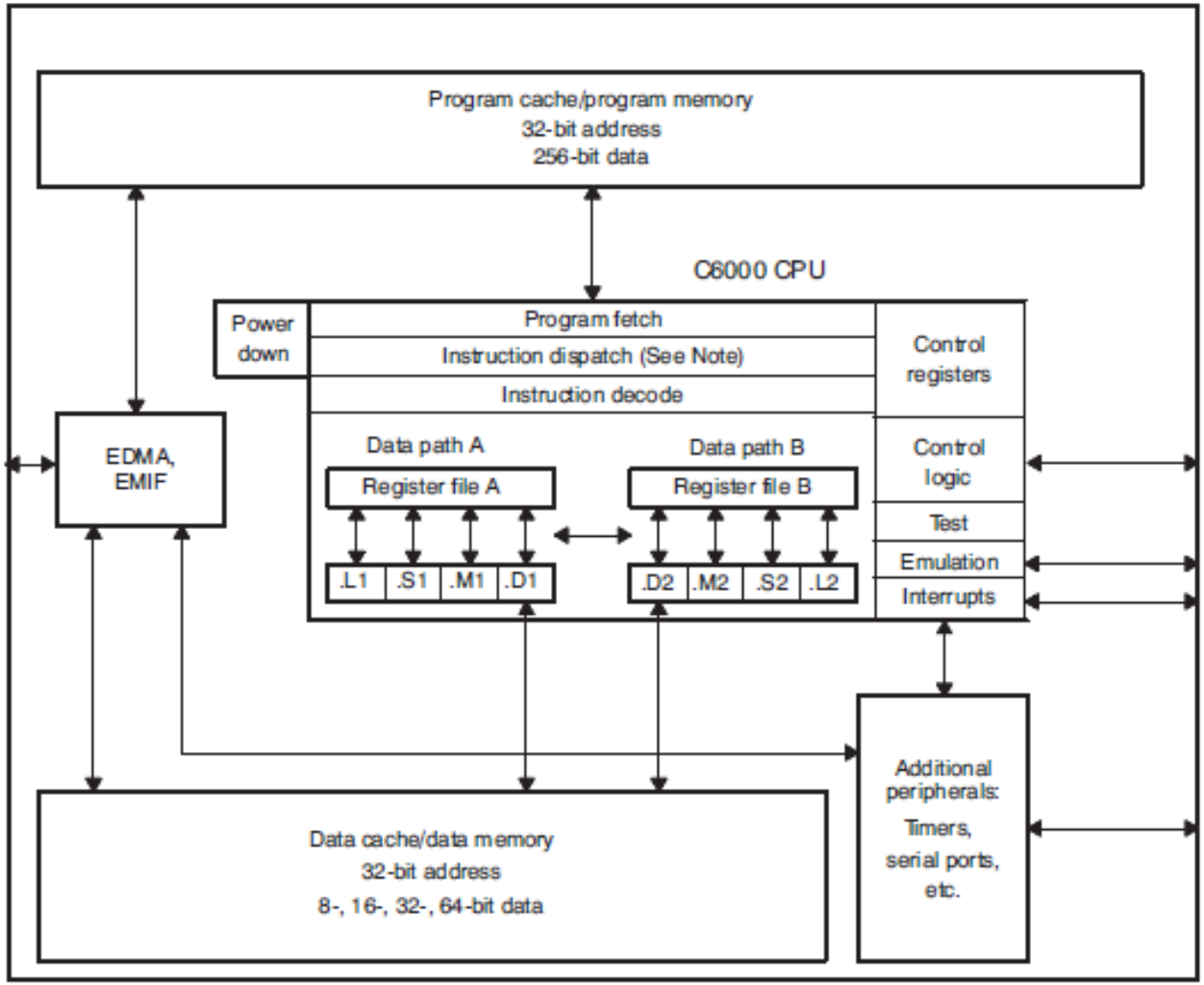
Разработка программного обеспечения для сигнальных процессоров TMS320C64xx

Часть 3. Архитектура ядра процессоров c64xx.

Ключевые характеристики ЦСП TMS320C64xx

- Архитектура VLIW, содержащая 8 функциональных блоков, позволяющая выполнять 8 инструкций за такт;
- Упаковка инструкций, способствующая уменьшению объема программного кода;
- Возможность условного выполнения большинства инструкций повышающая гибкость кода использующего ветвления;
- Поддержка 8/16/32 битных данных обеспечивающее эффективное использование памяти;
- Возможность использования 40 битых арифметических операций повышает точность вычислений.

Блок-схема ЦСП TMS320C64xx



Ядро C64xx содержит:

- модуль выборки инструкций;
- модуль диспетчера инструкций;
- модуль декодирования инструкций;
- 2 пути передачи данных, по одному на 4 функциональных блока;
- 64 32-битных регистра;
- регистры управления;
- логика управления;
- логика контроля, эмуляции, и прерываний.

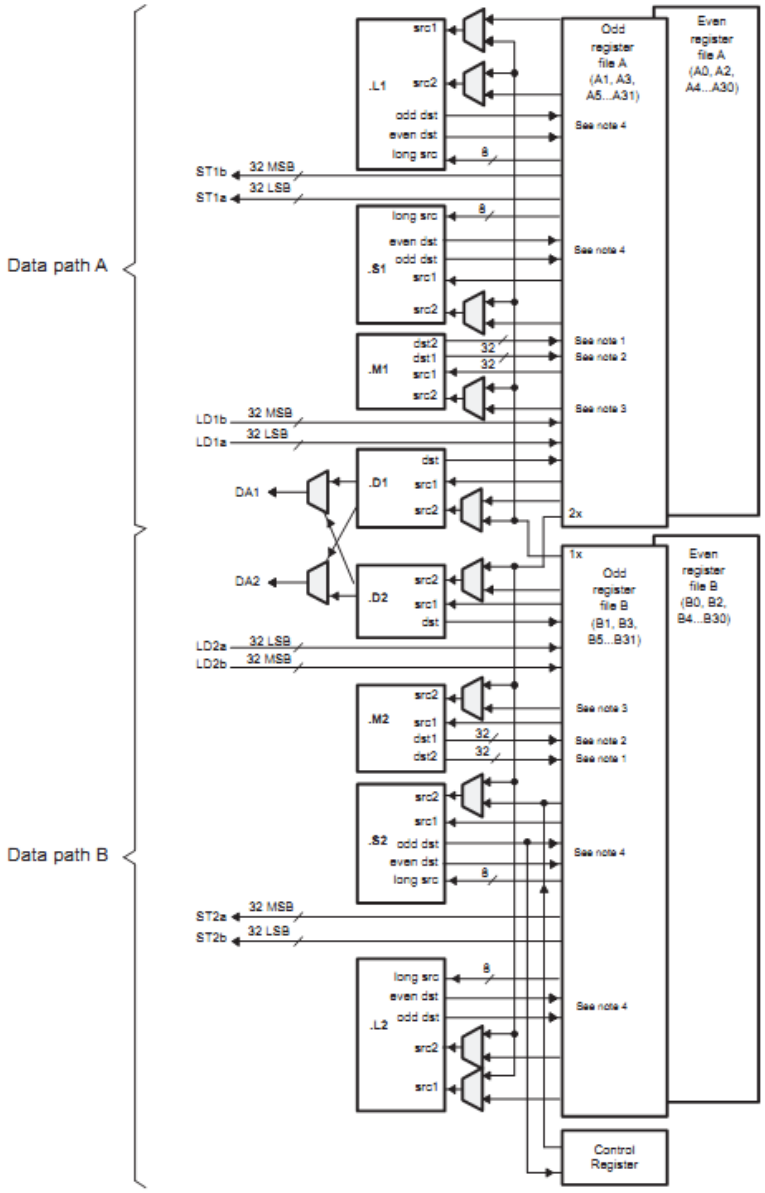
Внутренняя память ядра c64xx

- Память разделена на память программ и память данных и имеет 32-битную адресацию.
- Имеется 2 внутренних 64-битных порта для доступа к памяти данных.
- Имеется 1 внутренний 256-битный порт для доступа к памяти программ.

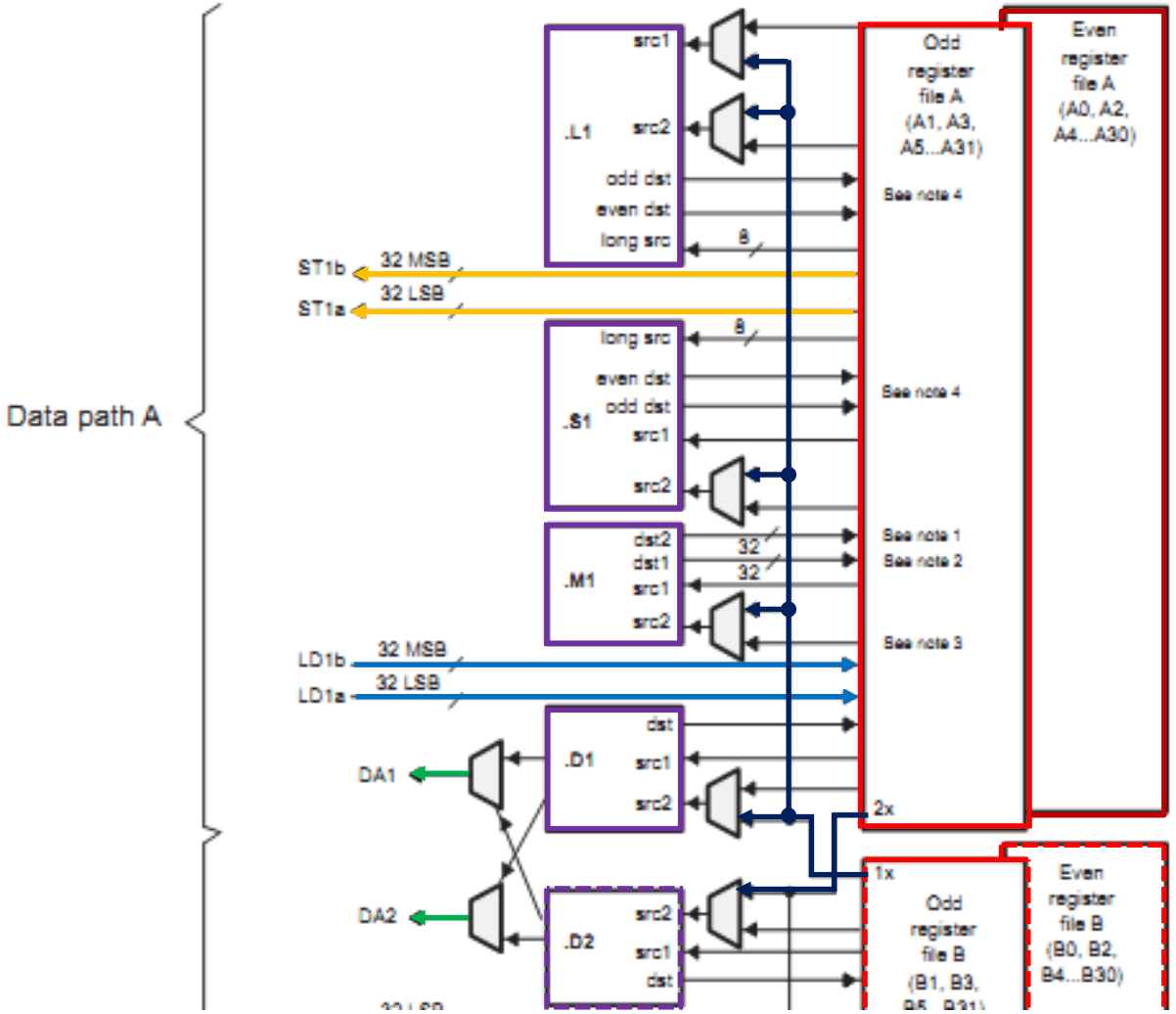
Также ЦСП C6400 содержат:

- Встроенную RAM объемом до 7 Mbit.
- 32-битный интерфейс внешней памяти (EMIF) поддерживающий SDRAM, SBSRAM, SRAM и другие типы асинхронной памяти;
- Контроллер ПДП (EDMA) имеющий 16 независимых каналов;
- Параллельный HPI интерфейс посредством которого можно осуществлять доступ в адресное пространство другого ЦСП;
- Несколько буферизованных последовательных портов McBSP;
- Как минимум 2 32-битных таймера.

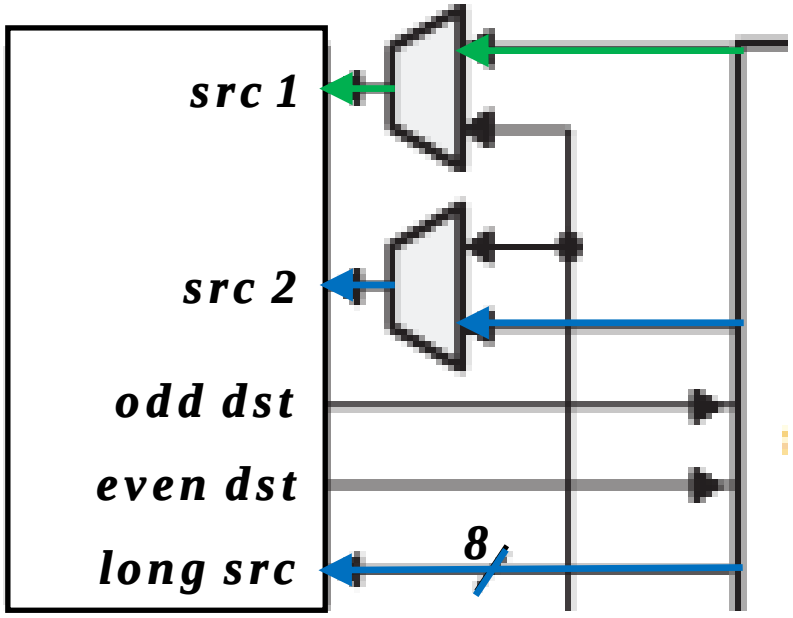
Компоненты путей данных ядра



Путь данных A

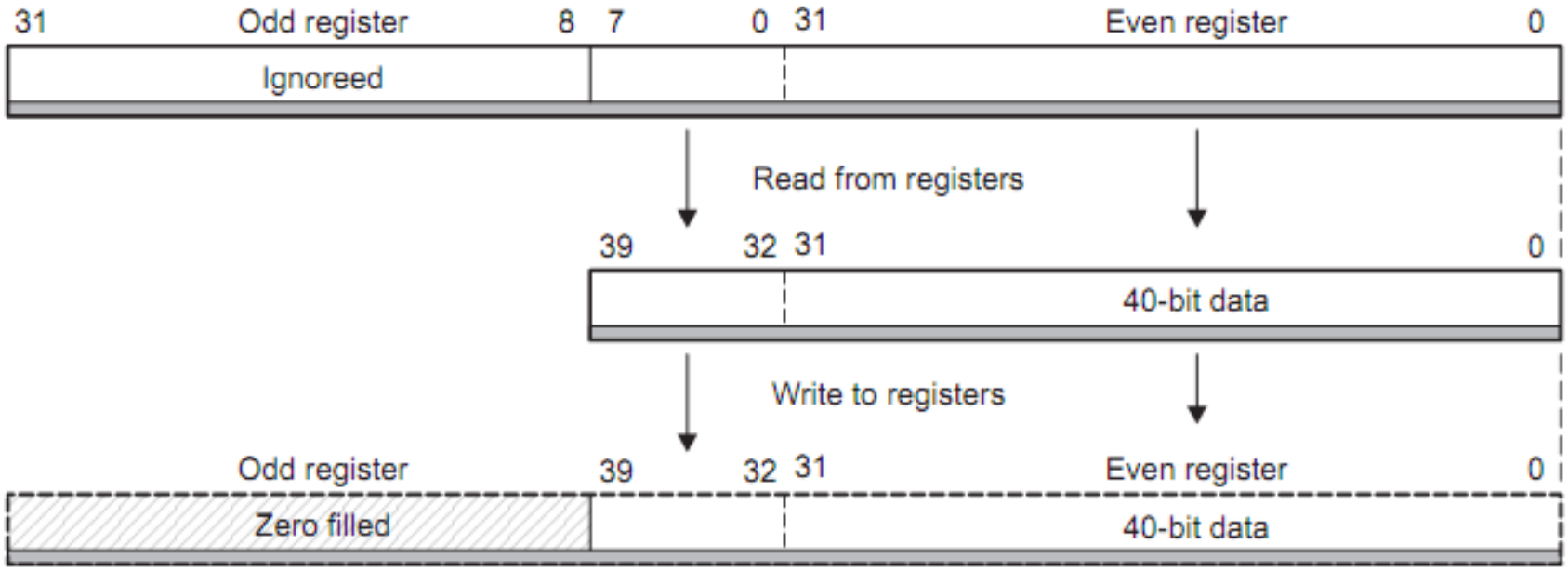


Модуль .L выполняет

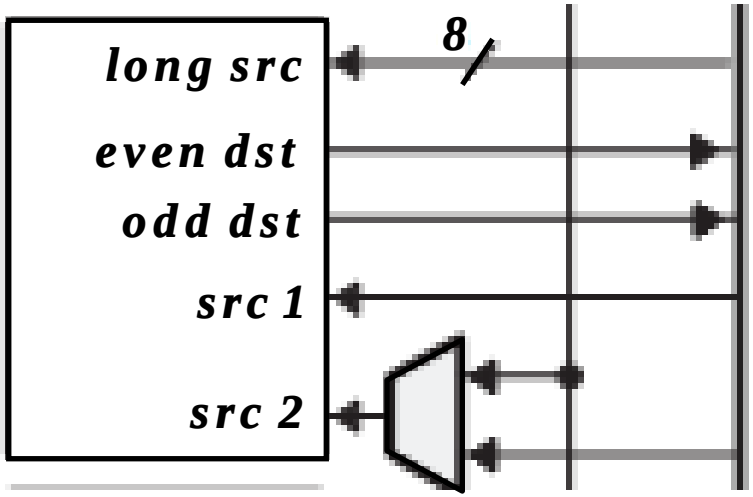


- 32/40-bit arithmetic and compare operations
- 32-bit logical operations
- Leftmost 1 or 0 counting for 32 bits
- Normalization count for 32 and 40 bits
- Byte shifts
- Data packing/unpacking
- 5-bit constant generation
- Dual 16-bit arithmetic operations
- Quad 8-bit arithmetic operations
- Dual 16-bit minimum/maximum operations
- Quad 8-bit minimum/maximum operations

Принцип размещения 40-битных данных в регистровой паре

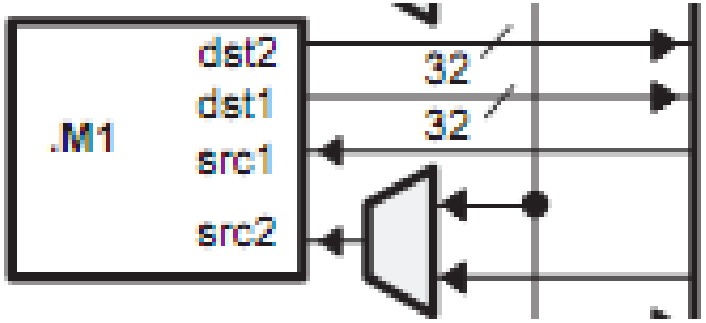


Модуль .S выполняет



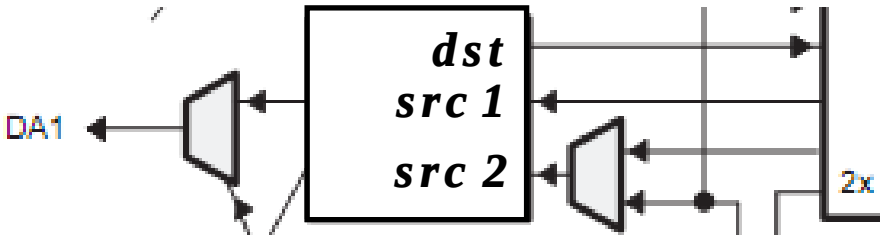
- 32-bit arithmetic operations
- 32/40-bit shifts and 32-bit bit-field operations
- 32-bit logical operations
- Branches
- Constant generation
- Register transfers to/from control register file (.S2 only)
- Byte shifts
- Data packing/unpacking
- Dual 16-bit compare operations
- Quad 8-bit compare operations
- Dual 16-bit shift operations
- Dual 16-bit saturated arithmetic operations
- Quad 8-bit saturated arithmetic operations

Модуль .M выполняет



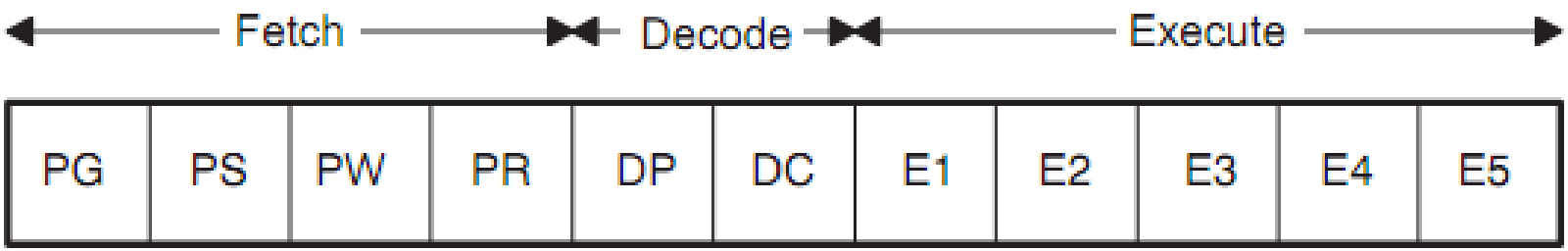
- 32 × 32-bit multiply operations
- 16 × 16-bit multiply operations
- 16 × 32-bit multiply operations
- Quad 8 × 8-bit multiply operations
- Dual 16 × 16-bit multiply operations
- Dual 16 × 16-bit multiply with add/subtract operations
- Quad 8 × 8-bit multiply with add operation
- Bit expansion
- Bit interleaving/de-interleaving
- Variable shift operations
- Rotation
- Galois Field Multiply

Модуль .D выполняет

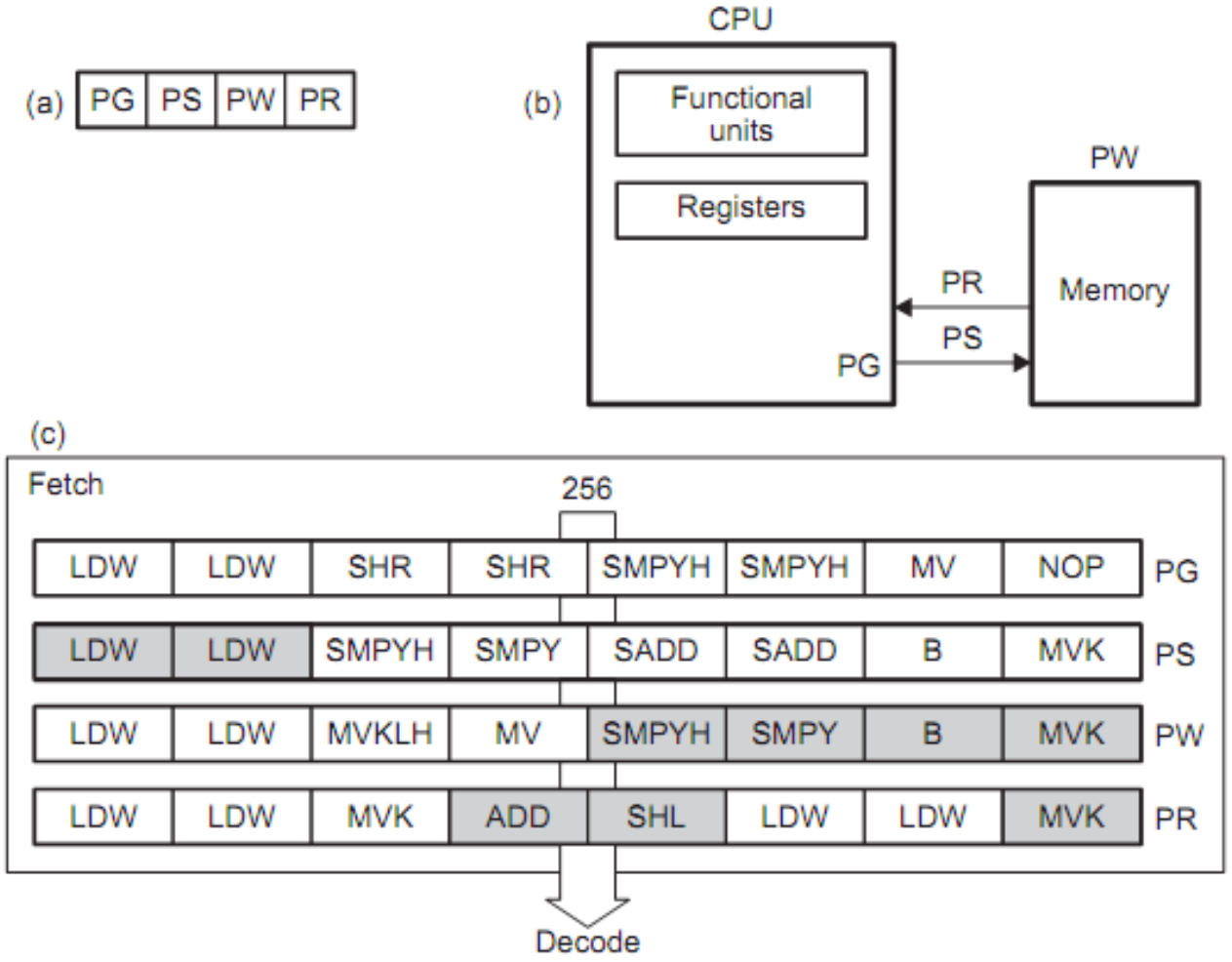


- 32-bit add, subtract, linear and circular address calculation
- Loads and stores with 5-bit constant offset
- Loads and stores with 15-bit constant offset (.D2 only)
- Load and store doublewords with 5-bit constant
- Load and store nonaligned words and doublewords
- 5-bit constant generation
- 32-bit logical operations

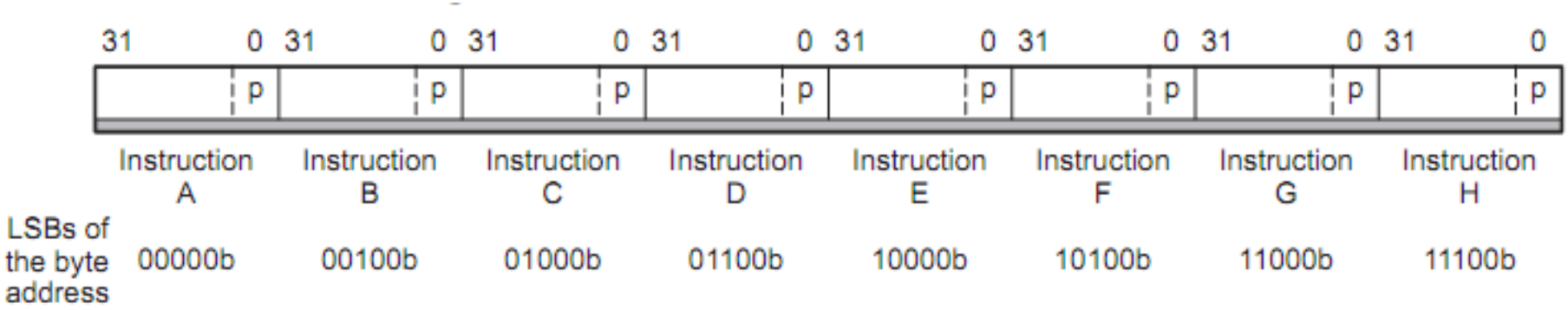
Стадии и фазы конвейера



Стадия выборки инструкций



Формат пакета выборки



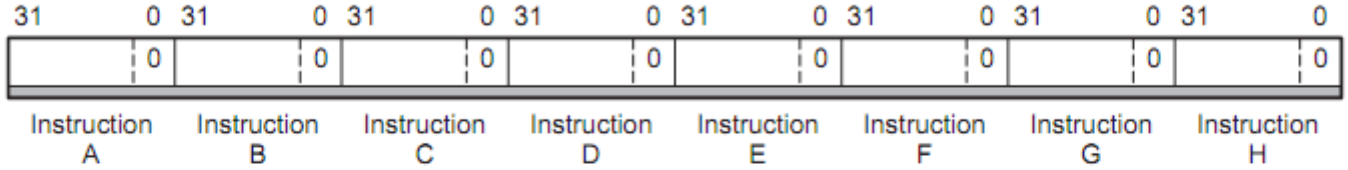
Пример записи инструкций выполняемых параллельно:

```
        instruction A

        instruction B

        instruction C
||      instruction D
||      instruction E

        instruction F
||      instruction G
||      instruction H
```

Пример пакета где все инструкции выполняются последовательно

The eight instructions are executed sequentially.

This *p*-bit pattern:



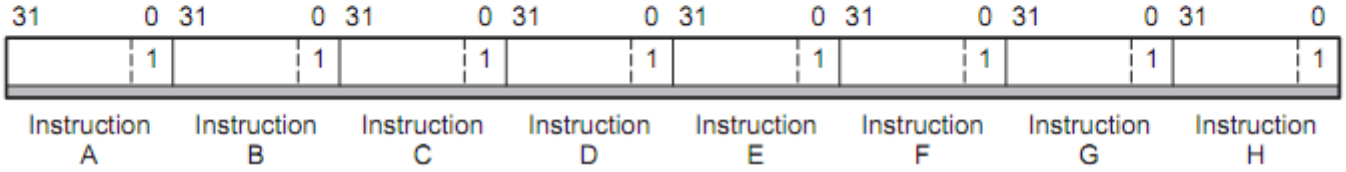
results in this execution sequence:

Cycle/Execute Packet	Instructions
1	A
2	B
3	C
4	D
5	E
6	F
7	G
8	H

Пример пакета где все инструкции выполняются параллельно

All eight instructions are executed in parallel.

This *p*-bit pattern:

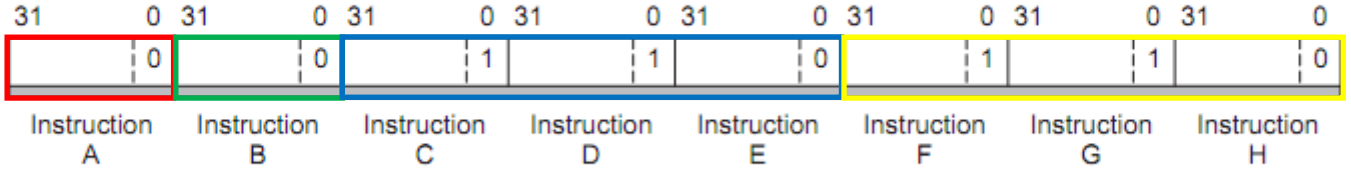


results in this execution sequence:

Cycle/Execute Packet	Instructions							
1	A	B	C	D	E	F	G	H

Пример пакета с последовательно-параллельным выполнением инструкций

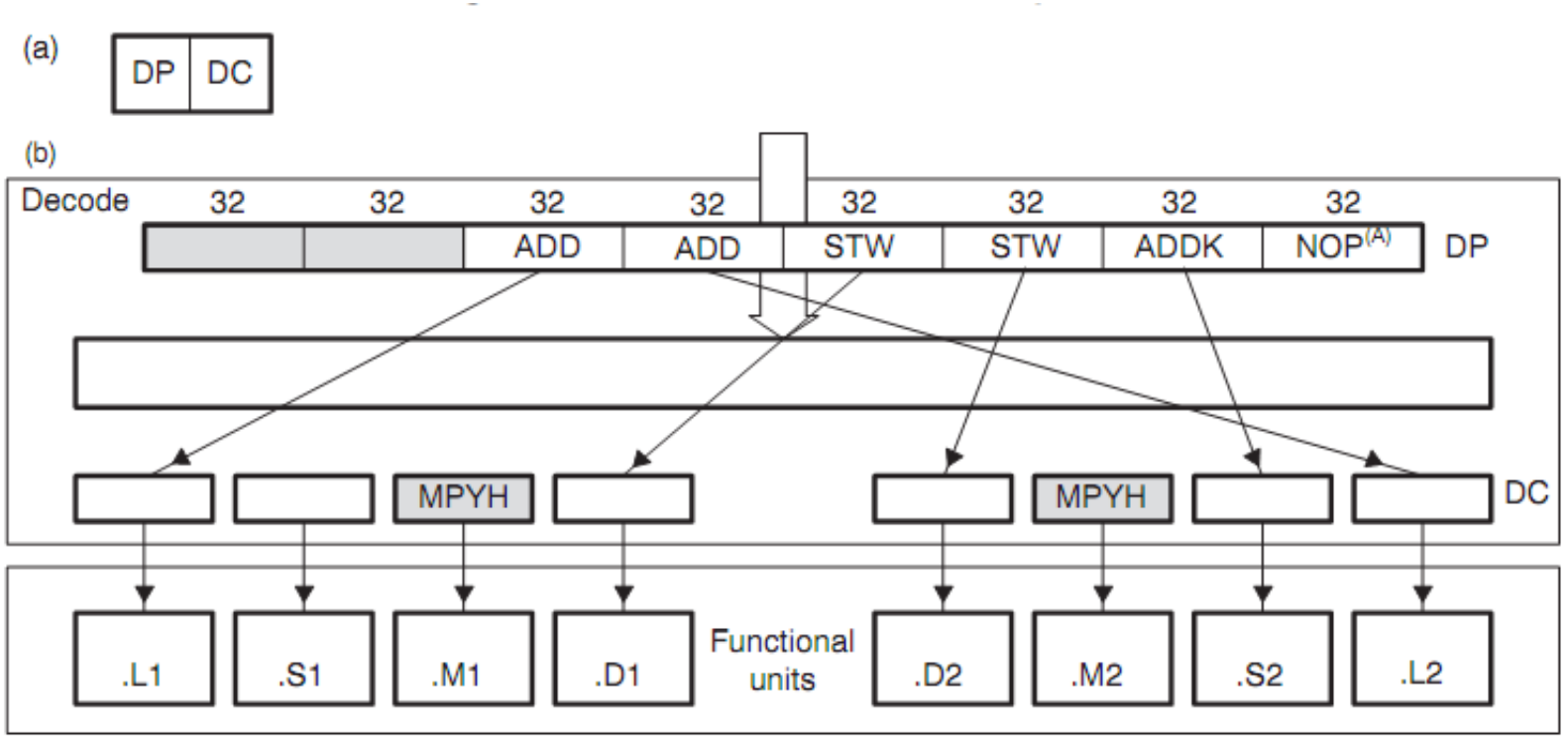
This *p*-bit pattern:



results in this execution sequence:

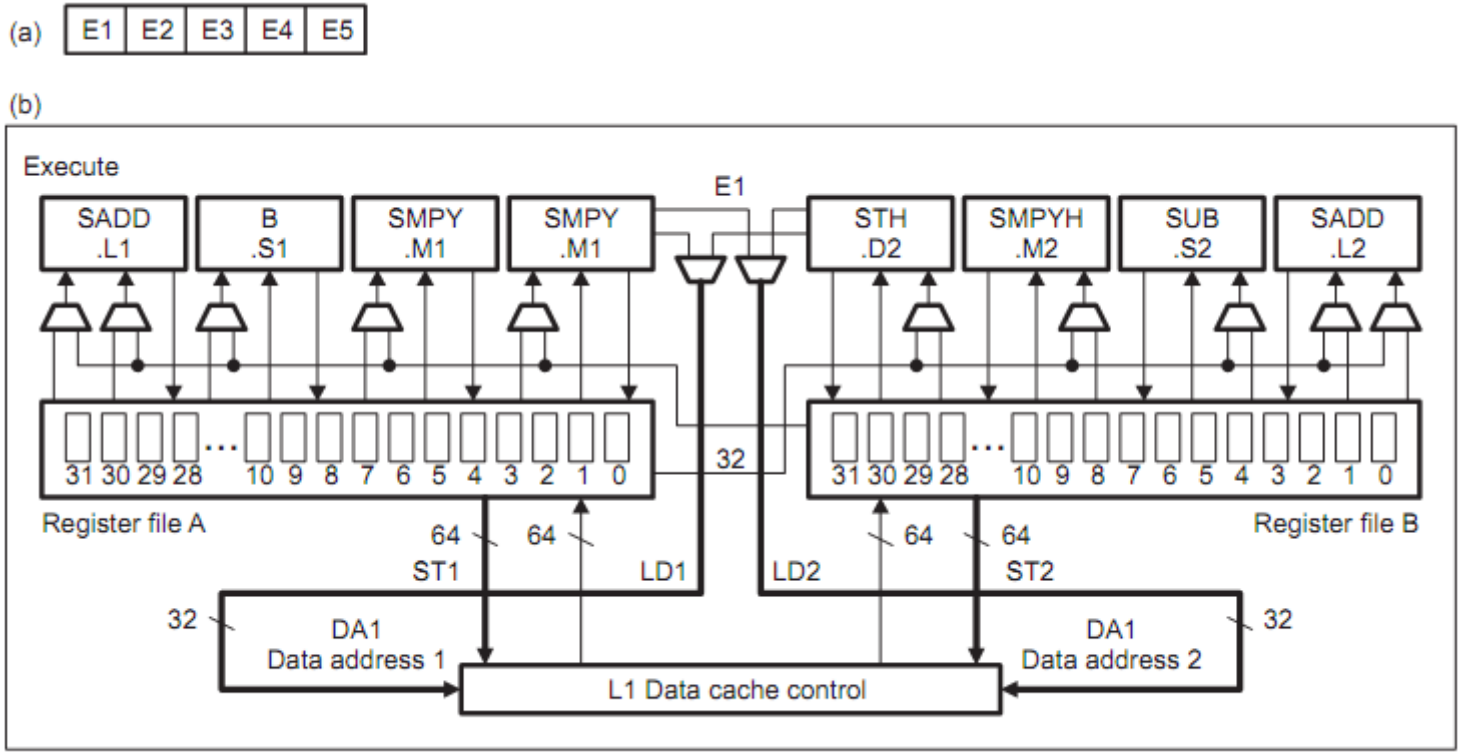
Cycle/Execute Packet	Instructions		
1	A		
2	B		
3	C	D	E
4	F	G	H

Стадия диспетчеризации и декодирования инструкций



A NOP is not dispatched to a functional unit.

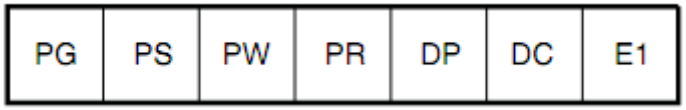
Стадия выполнения инструкций



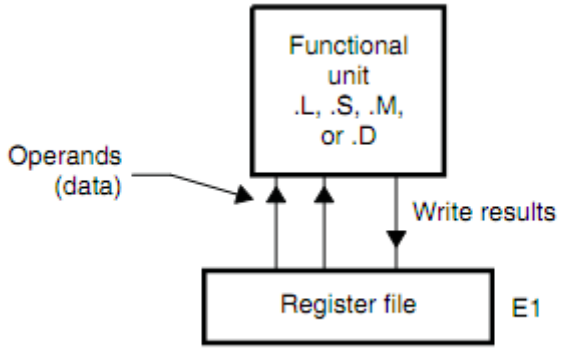
Фазы последней стадии конвейера для инструкции выполняемой за 1 такт

Pipeline Stage	E1
Read	*src1, src2*
Written	*dst*
Unit in use	.L, .S, .M, or .D

Фазы выполнения инструкций:



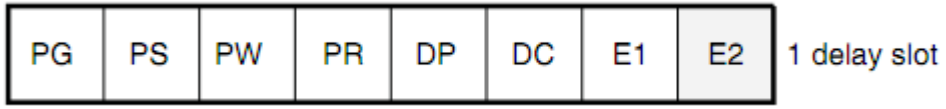
Блок схема стадии выполнения:



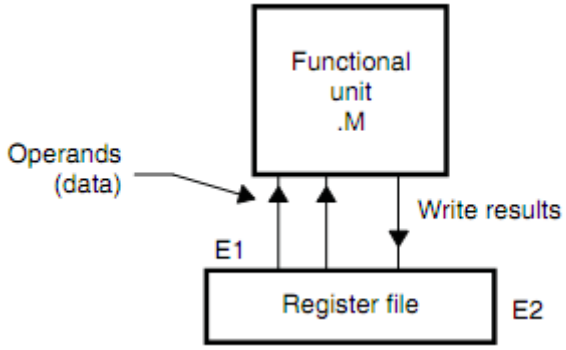
Фазы последней стадии конвейера для инструкции выполняемой за 2 такта

Pipeline Stage	E1	E2
Read	*src1, src2*	
Written		*dst*
Unit in use	.M	

Фазы выполнения инструкций:



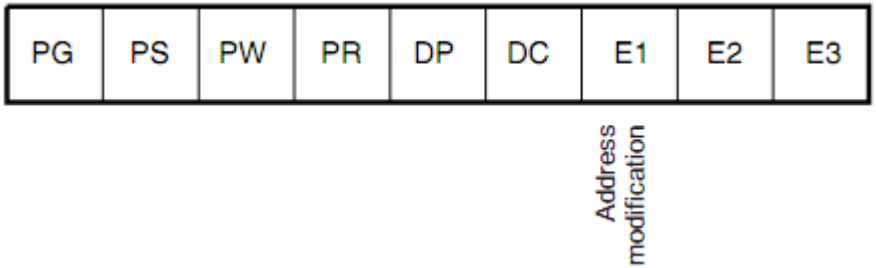
Блок схема стадии выполнения:



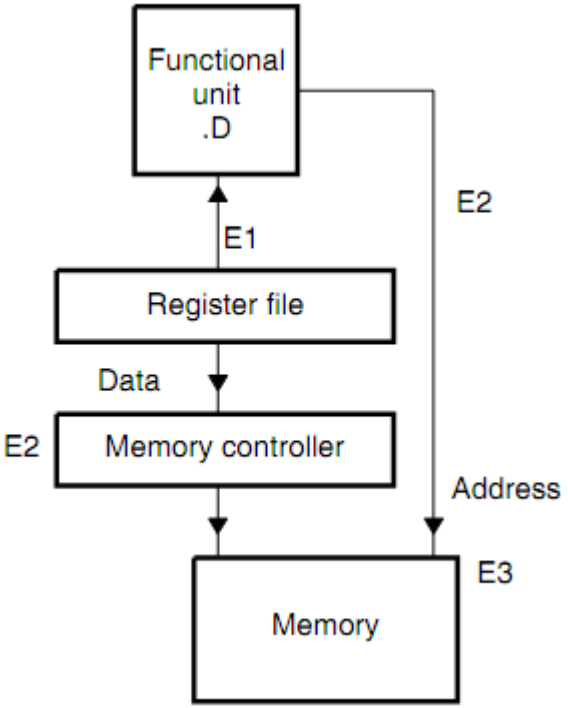
Фазы последней стадии конвейера для инструкции выполняемой за 3 такта

Pipeline Stage	E1	E2	E3
Read	*baseR, offsetR, src*		
Written	*baseR*		
Unit in use	.D2		

Фазы выполнения инструкций:



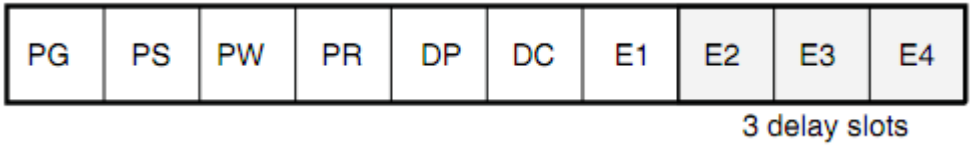
Блок схема стадии выполнения:



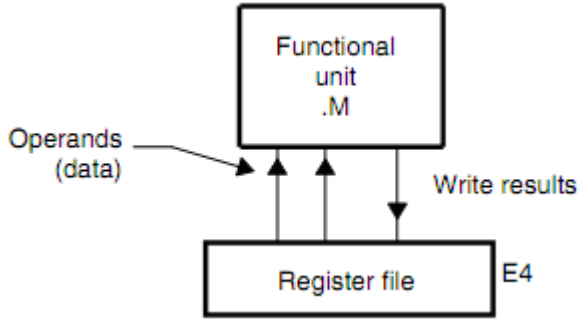
Фазы последней стадии конвейера для инструкции выполняемой за 4 такта

Pipeline Stage	E1	E2	E3	E4
Read	*src1, src2*			
Written				dst
Unit in use	.M			

Фазы выполнения инструкций:



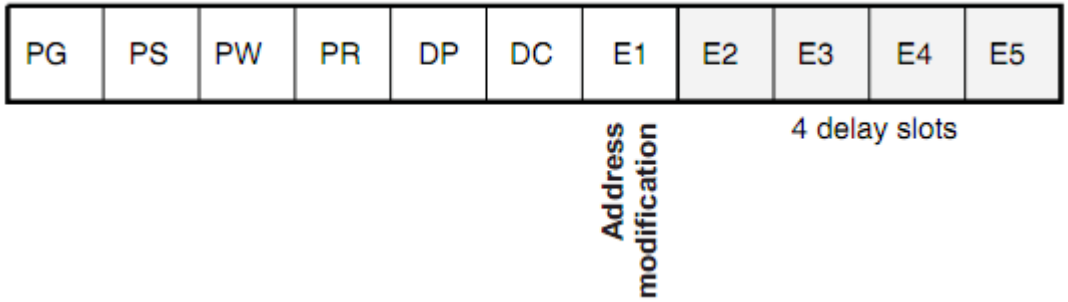
Блок схема стадии выполнения:



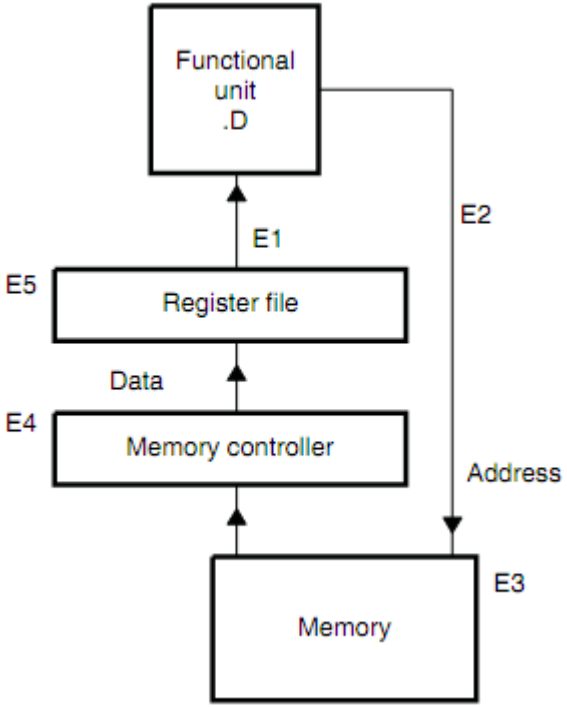
Фазы последней стадии конвейера для инструкции выполняемой за 5 тактов

Pipeline Stage	E1	E2	E3	E4	E5
Read	*baseR, offsetR, src*				
Written	*baseR*				*dst*
Unit in use	.D				

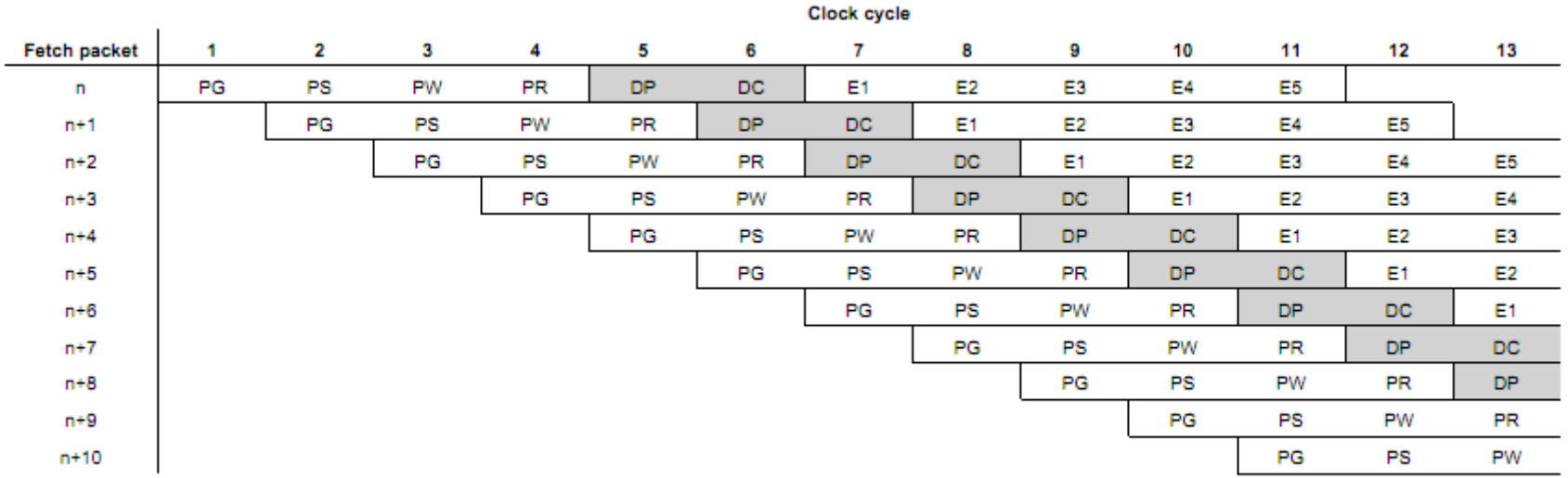
Фазы выполнения инструкций:



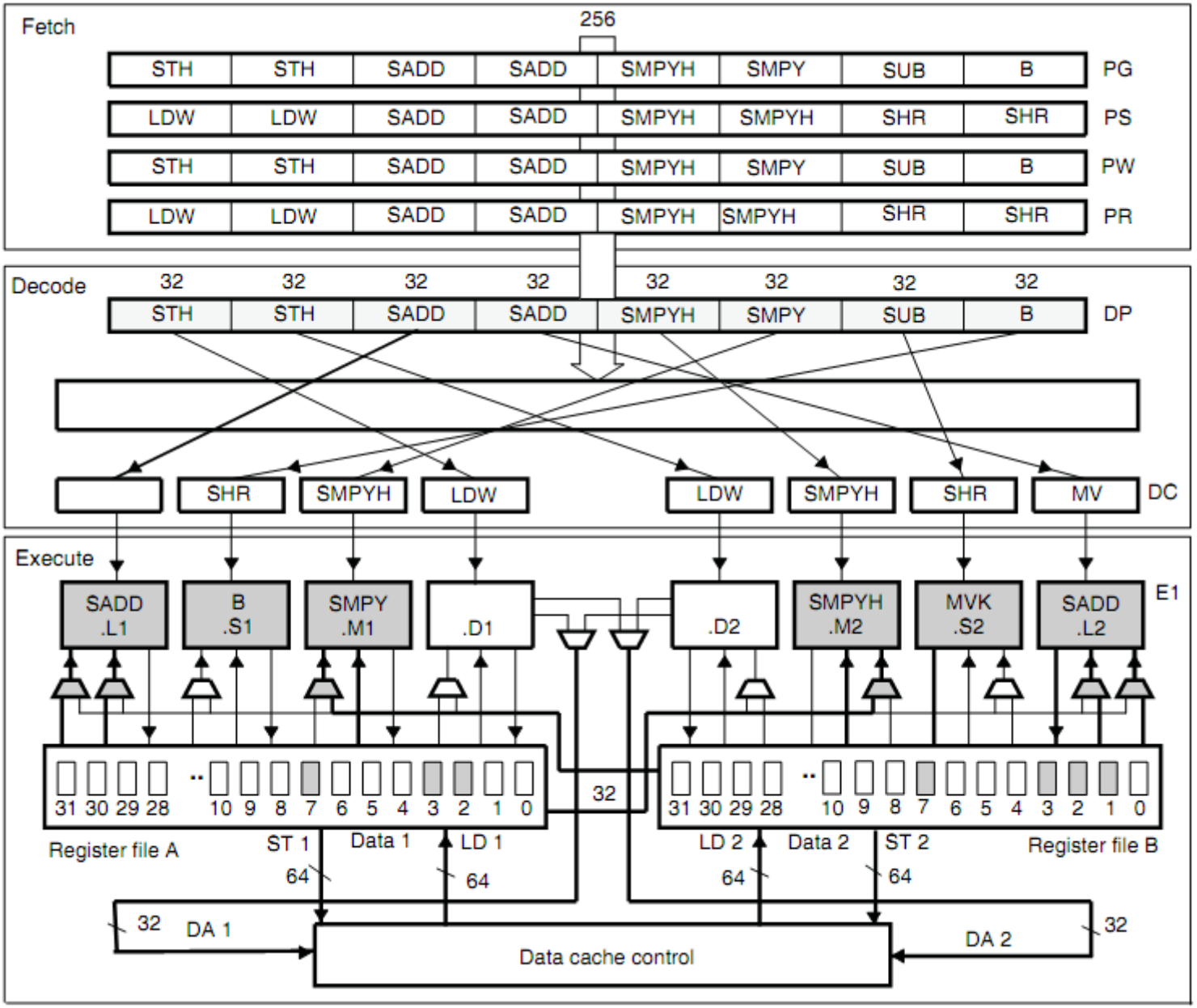
Блок схема стадии выполнения:



Временная диаграмма работы конвейера



* Случай когда пакет выборки содержит только один исполняемый пакет


Fetch
256
STH STH SADD SADD SMPYH SMPY SUB B PG
LDW LDW SADD SADD SMPYH SMPYH SHR SHR PS
STH STH SADD SADD SMPYH SMPY SUB B PW
LDW LDW SADD SADD SMPYH SMPYH SHR SHR PR
Decode
32 32 32 32 32 32 32 32
STH STH SADD SADD SMPYH SMPY SUB B DP
SHR SMPYH LDW LDW SMPYH SHR MV DC
Execute
SADD .L1
B .S1
SMPY .M1
.D1
.D2
SMPYH .M2
MVK .S2
SADD .L2
E1
31 30 29 28 .. 10 9 8 7 6 5 4 3 2 1 0
Register file A
ST 1
Data 1
LD 1
64
64
32
LD 2
Data 2
ST 2
64
64
Register file B
32
DA 1
Data cache control
DA 2
32

Разработка программного обеспечения для сигнальных процессоров TMS320C64xx

Часть 4. Прерывания.

Введение

Обычно, ЦСП работает с другими устройствами (периферией), которые формируют многочисленные асинхронные внешние сигналы (события), на которые процессор должен отреагировать определенным образом (обработать).

Прерывания – сигнал, сообщающий процессору о совершении какого-либо асинхронного события. При этом выполнение текущей последовательности команд приостанавливается, и управление передаётся обработчику прерываний, который выполняет работу по обработке события и возвращает управление в прерванный код.

Виды прерываний:

- аппаратные (внешние/внутренние)
- программные

Типы прерываний

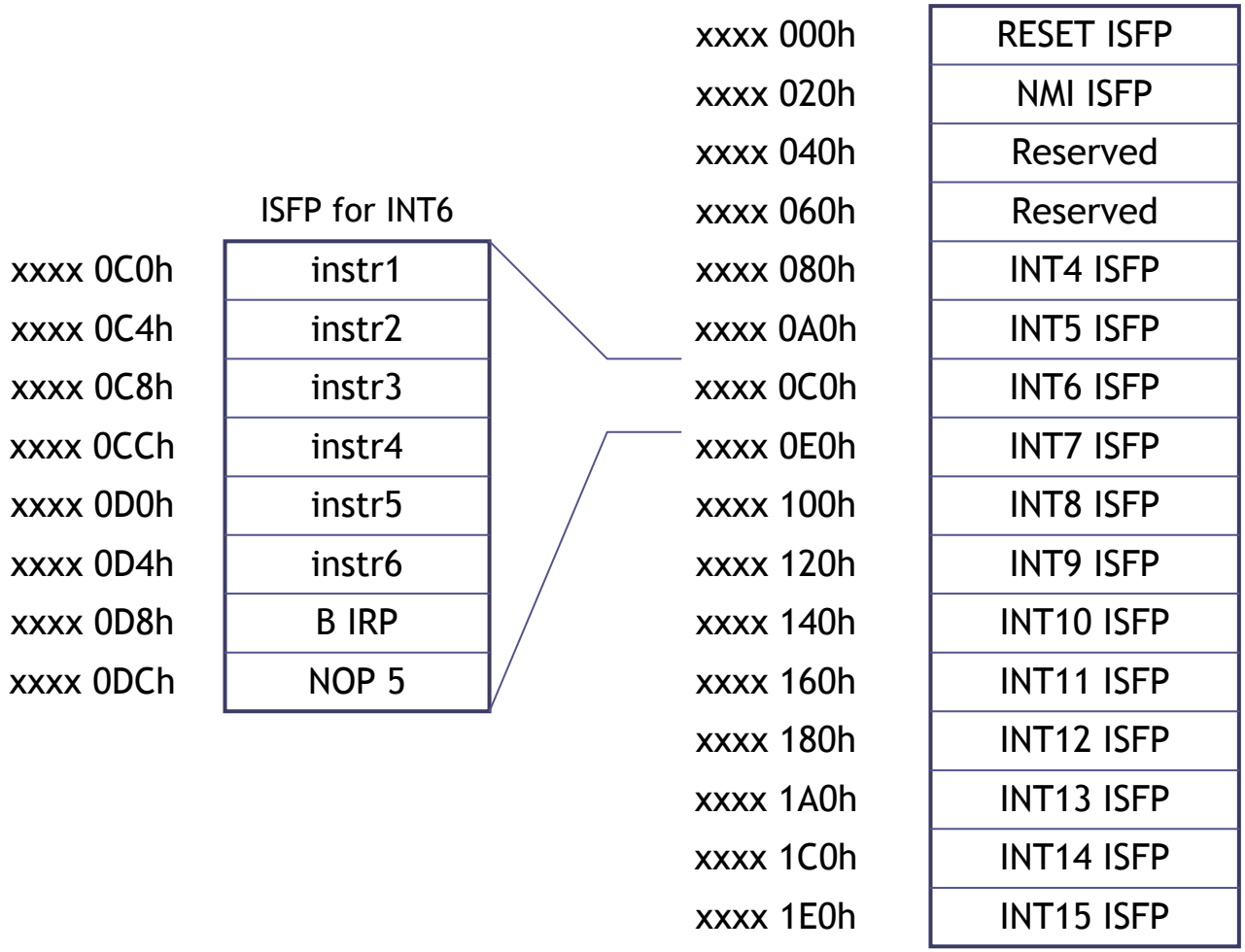
- RESET – имеет наивысший приоритет. Обрабатывается по приходу одноименного сигнала. Используется для остановки процессора и возврата в начальное известное состояние.
- Немаскируемое (NMI)– 2е по приоритету. Используется для оповещения процессора о серьезных неполадках в системе (как Exception).
- Маскируемые INT4 ..INT15 - могут быть связаны с внешними устройствами, встроенной периферией, решением программных задач.
- Исключение (поддерживается только ядром C64+)

Таблица приоритетов прерываний

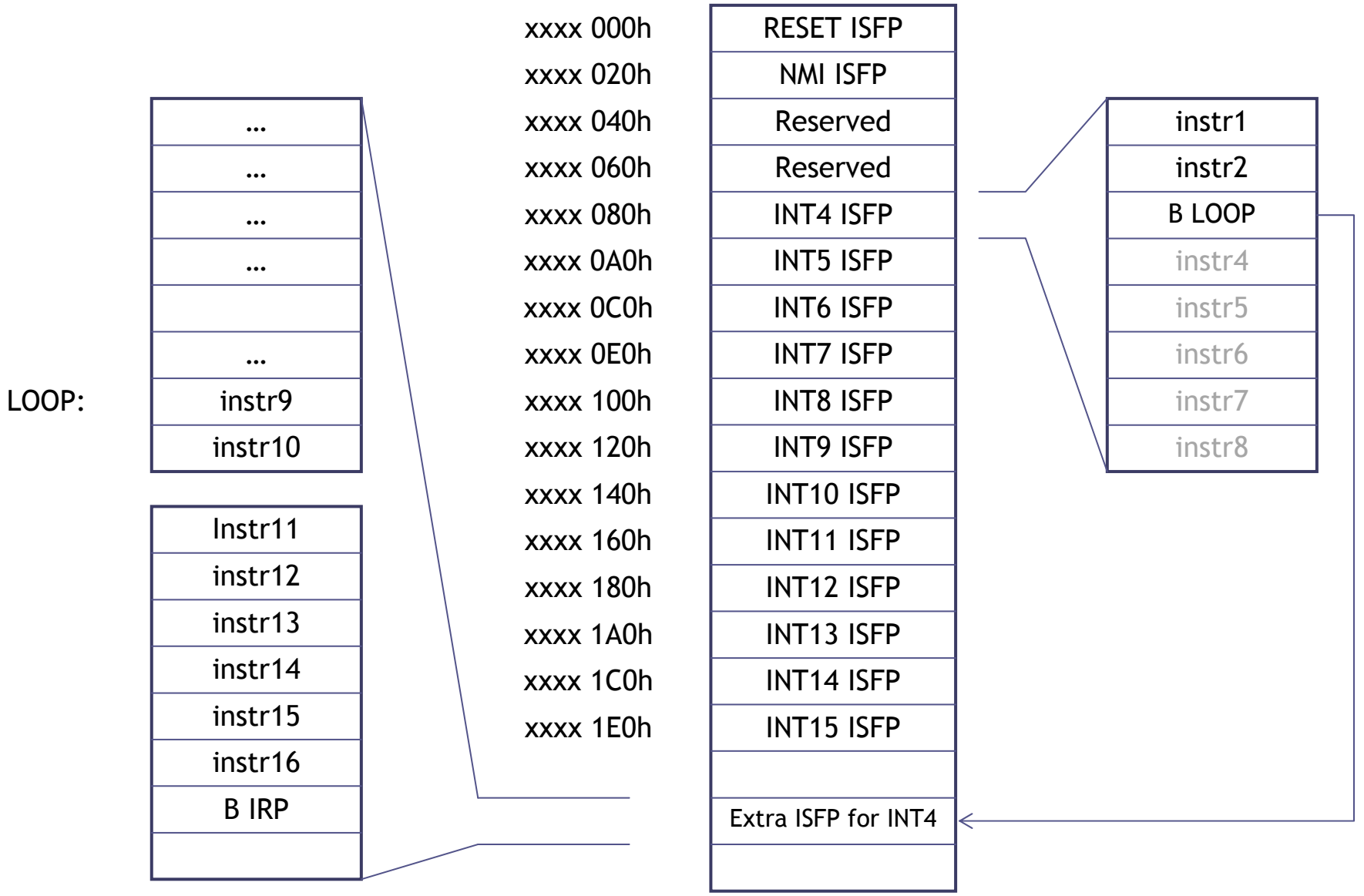
Приоритет	**Имя Прерывания**	**Тип Прерывания**
Высший	Reset	Reset
	NMI	Немаскируемое
	INT4	Маскируемое
	INT5	Маскируемое
	INT6	Маскируемое
	INT7	Маскируемое
	INT8	Маскируемое
	INT9	Маскируемое
	INT10	Маскируемое
	INT11	Маскируемое
	INT12	Маскируемое
	INT13	Маскируемое
	INT14	Маскируемое
Низший	INT15	Маскируемое

Таблица векторов прерываний

При возникновении прерывания, процессор обращается к таблице векторов прерываний. В таблице расположены обработчики прерываний, каждый из которых содержит 8 32-разрядных инструкций.



Обработчик прерывания



Регистры, используемые при работе с прерываниями

Аббр.	Имя регистра	Описание
CSR	Control status register	Позволяет разрешить/запретить прерывания
ICR	Interrupt clear register	Позволяет вручную сбрасывать флаги в IFR
IER	Interrupt enable register	Позволяет разрешить прерывания
IFR	Interrupt flag register	Показывает статус прерываний
IRP	Interrupt return pointer register	Содержит адрес возврата, используемый при возвращении из маскируемого прерывания.
ISR	Interrupt set register	Позволяет вручную выставлять флаги в IFR
ISTP	Interrupt task state register	Указатель на начало таблицы обработчика прерываний
ITSR	Interrupt task state register	Interrupted (non-NMI) machine state.(C64x+ CPU only)
NRP	Nonmaskable interrupt return pointer register	Содержит адрес возврата из немаскируемого прерывания
TSR	Task state register	Позволяет глобально разрешать или запрещать прерывания.

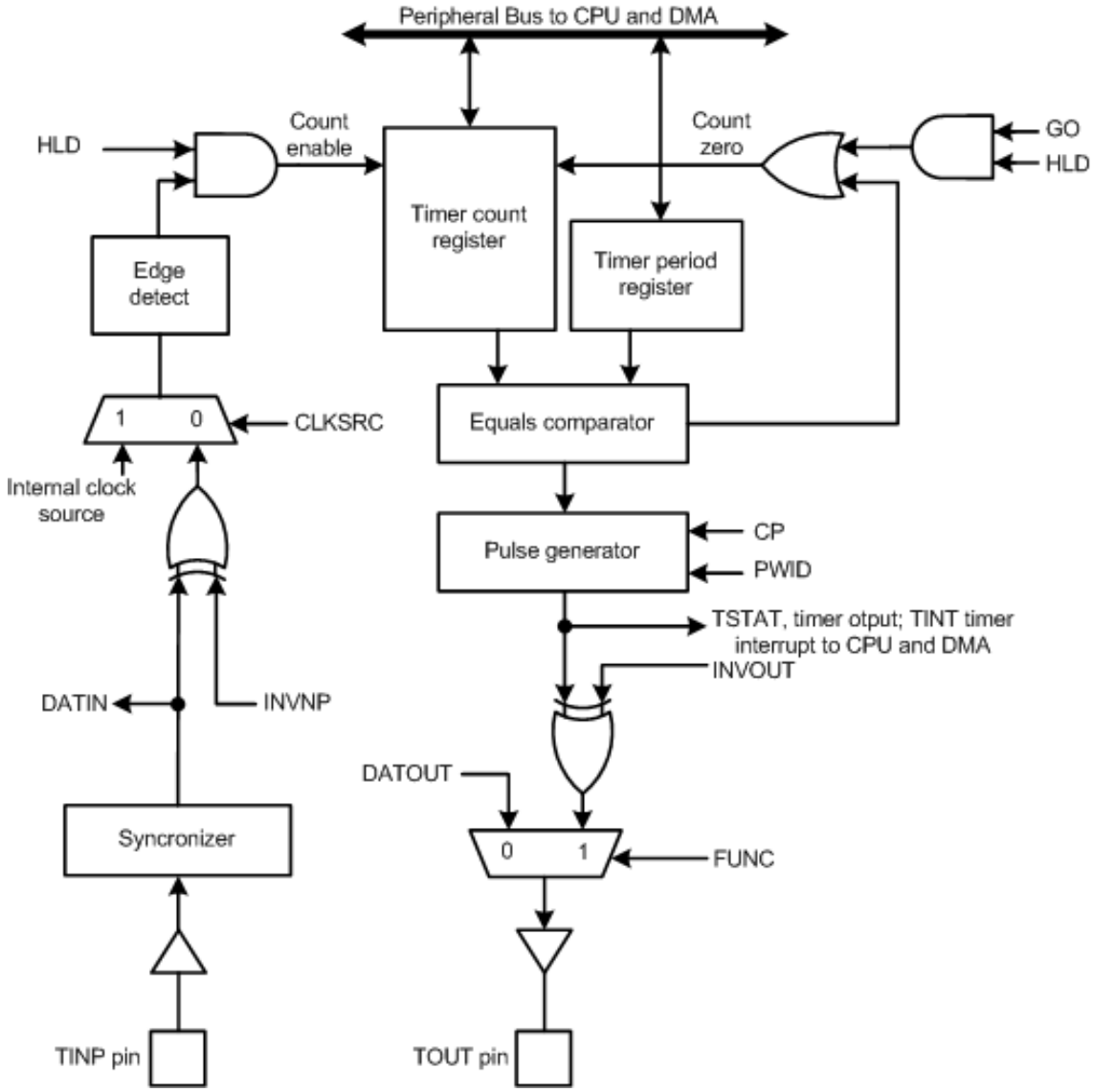
Разработка программного обеспечения для сигнальных процессоров TMS320C64xx

Часть 5. Таймер. McBSP.

Назначение таймера

- Отсчет временных интервалов
- Подсчет числа импульсов
- Генерация импульсов
- Генерация прерываний для CPU
- Посылка событий синхронизации контроллеру DMA

Структурная схема таймера



Регистры таймера

Timer Control Register (CTL)

31							16
Reserved							
R-0							
15	14		12	11	10	9	8
SPND	Reserved			TSTAT	INVINP	CLKSRC	CP
R/W-0	R-0			R-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
HLD	GO	Reserved	PWID	DATIN	DATOUT	INVOUT	FUNC
R/W-0	R/W-0	R-0	R/W-0	R-x	R/W-0	R/W-0	R/W-0

LEGEND: R = Read only; R/W = Read/Write; -n = value after reset; -x = value is indeterminate after reset

Timer Period Register (PRD)

31	0
Timer Period (PRD)	
R/W-0	

LEGEND: R/W = Read/Write; -n = value after reset

Timer Count Register (CNT)

31	0
Timer Count (CNT)	
R/W-0	

LEGEND: R/W = Read/Write; -n = value after reset

Порядок сброса и запуска таймера

Операция	GO	HLD	Описание
Останов таймера	0	0	Счет остановлен
Перезапуск	0	1	Таймер продолжает счет с значения до останова. Счетчик таймера не сбрасывается.
Запуск таймера	1	1	Счетчик таймера сбрасывается в 0 и запускается счет. Бит GO после этого самобонуляется.

- Если таймер не остановлен – останавливаем счет, записав в HLD 0;
- Записываем необходимое значение в регистр PRD;
- Задаем конфигурацию таймера записав нужное значение в регистр CTL, биты GO и HLD при этом не должны быть изменены;
- Запускаем таймер установив биты GO и HLD в 1.

Временные диаграммы

Figure 2. Timer Operation in Pulse Mode (CP = 0)

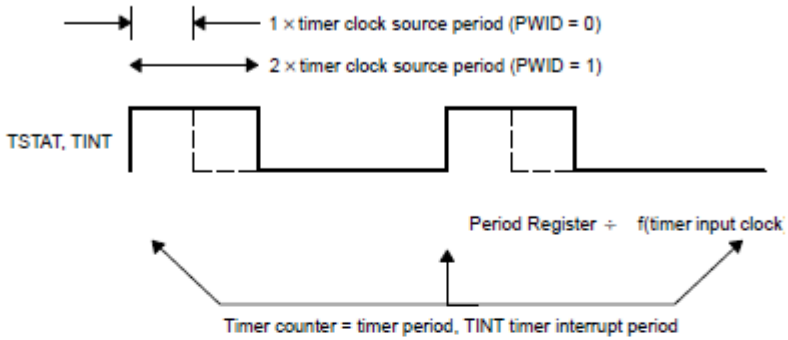
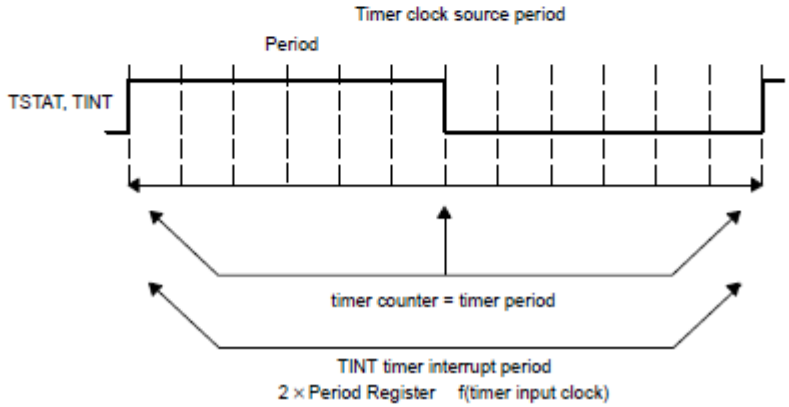


Figure 3. Timer Operation in Clock Mode (CP = 1)



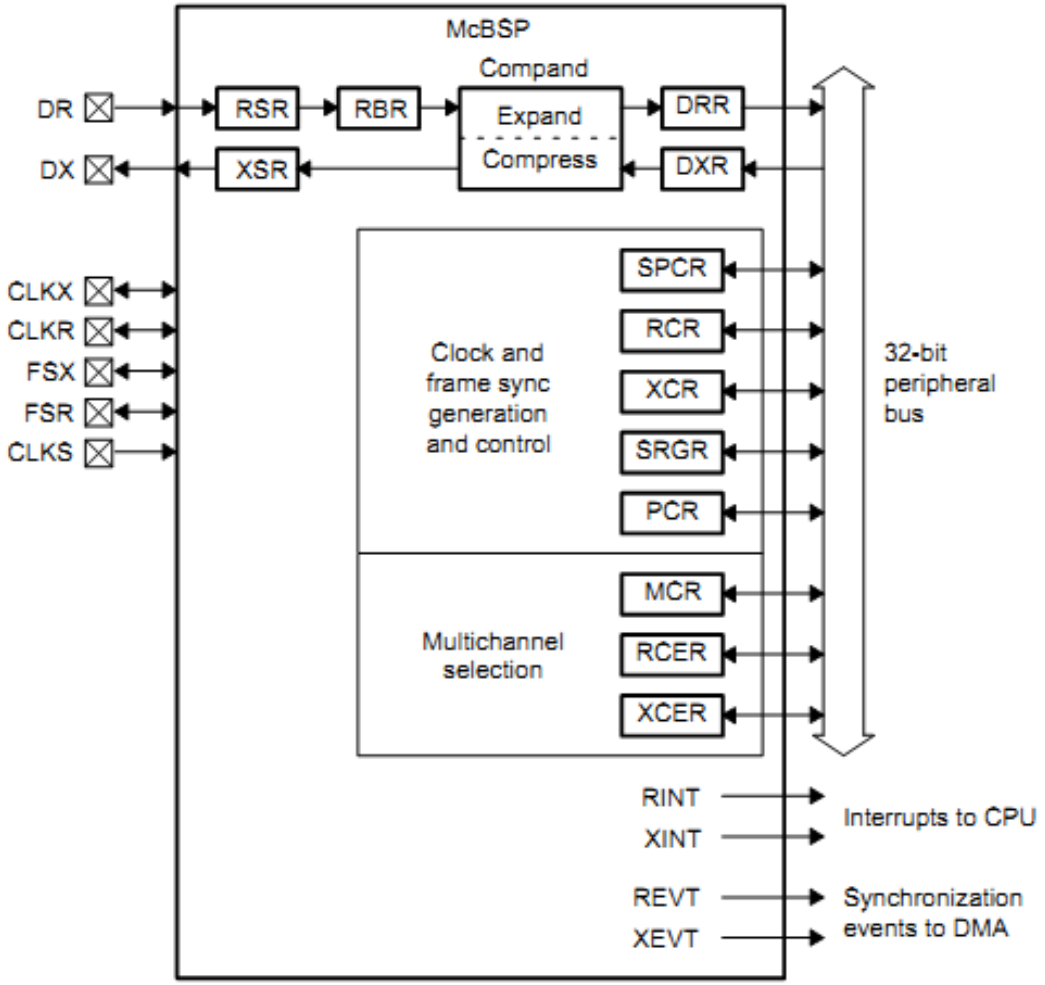
Временные параметры

Режим	Частота	Период	Длительность «1»	Длительность «0»
Pulse	f(clock source)/PRD	PRD/f(clock source)	(PWID+1)/ f(clock source)	PRD – (PWID+1)/f(clock source)
Clock	f(clock source)/2*PRD	2*PRD/f(clock source)	PRD/f(clock source)	PRD/f(clock source)

Мультиканальный буферизованный последовательный порт (McBSP)

- Полнодуплексный режим работы;
- Двойная буферизация регистров данных, позволяющая обеспечить непрерывный поток данных;
- Независимое тактирование линий передачи и приема;
- Прямое подключение к индустриально-стандартным кодекам, ЦАП, АЦП и AIC имеющим последовательный интерфейс;
- Прямое подключение к AC97, IIS, SPI совместимым устройствам;
- Возможность мультиканальной передачи и приема данных (до 128 каналов);
- Возможность выбора размера передаваемых данных (поддерживаются 8, 12, 16, 20, 24 и 32 битные данные);
- Для 8-битных данные имеется возможность выбора порядка передачи битов (LSB или MSB);
- Возможность программирования полярности тактов.

Структурная схема McBSP



Описание регистра SPCR

31						26	25	24
Reserved							FREE(1)	SOFT(1)
R-0							R/W-0	R/W-0
23	22	21			20	19	18	17	16
FRST	GRST	XINTM				XSYNCERR	XEMPTY	XRDY	XRST
R/W-0	R/W-0	R/W-0				R/W-0	R-0	R-0	R/W-0
15	14			13	12		11	10			8
DLB	RJUST				CLKSTP			Reserved			
R/W-0	R/W-0				R/W-0			R-0			
7	6	5			4	3	2	1	0
DXENA(1)	Reserved	RINTM				RSYNCERR	RFULL	RRDY	RRST
R/W-0	R-0	R/W-0				R/W-0	R-0	R-0	R/W-0

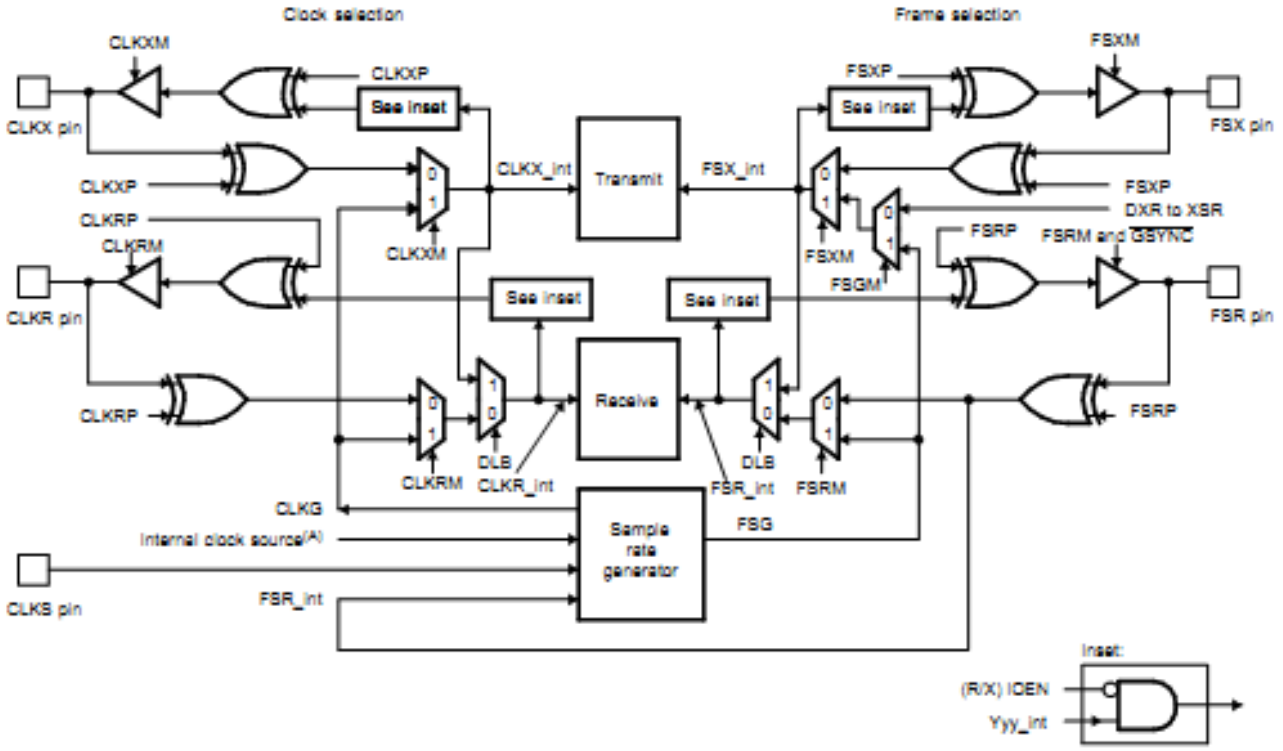
LEGEND: R = Read only; R/W = Read/ Write; -n = value after reset

Описание регистра RCR

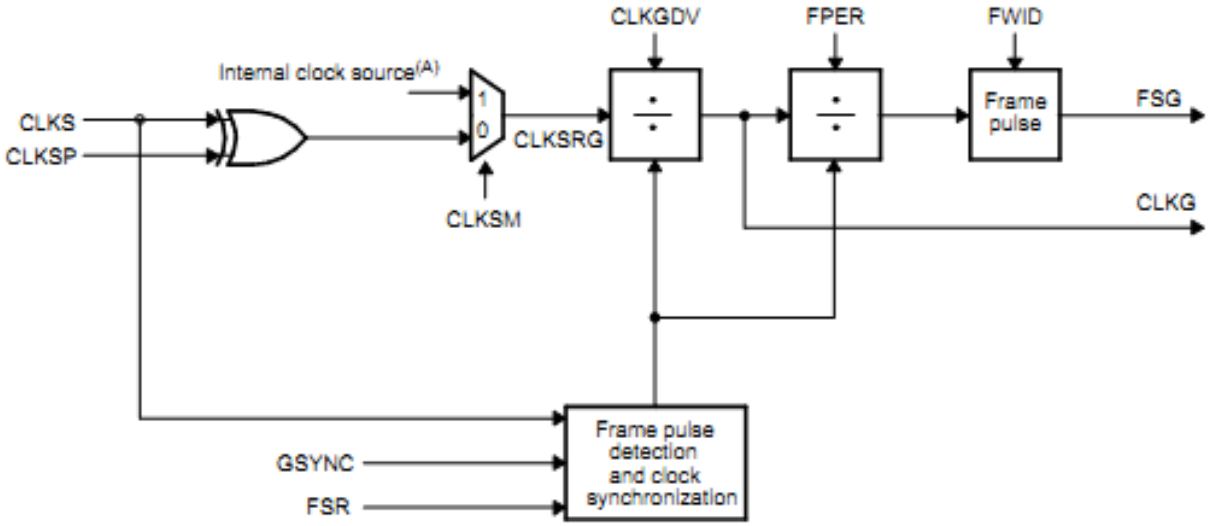
31	30 – 24	23 – 21	20 – 19	18	17 – 16
RPHASE	RFRLEN2	RWDLEN2	RCOMPAND	RFIG	RDATDLY
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

15	14 – 8	7 – 5	4	3 – 0
Reserved	RFRLEN1	RWDLEN1	RWDREVRS[1]	Reserved
R-0	R/W-0	R/W-0	R/W-0	R-0

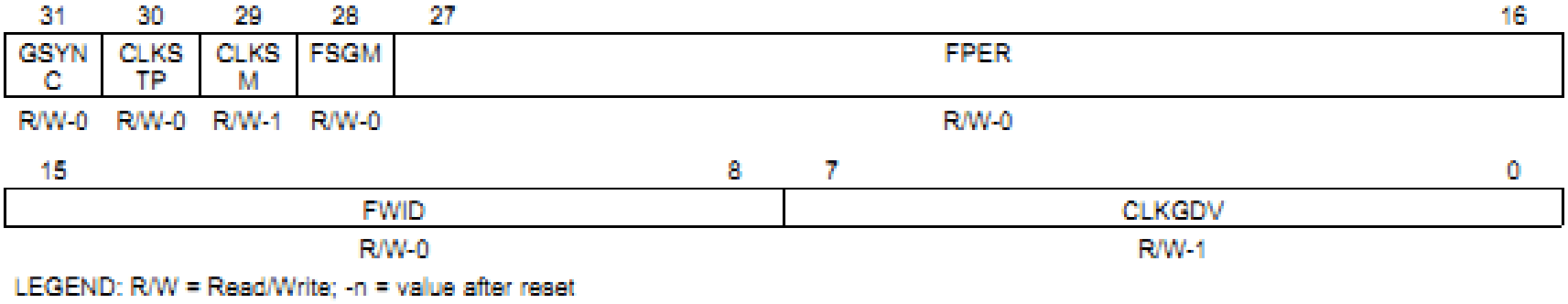
Принцип генерации тактов



Функциональная схема генератора тактов



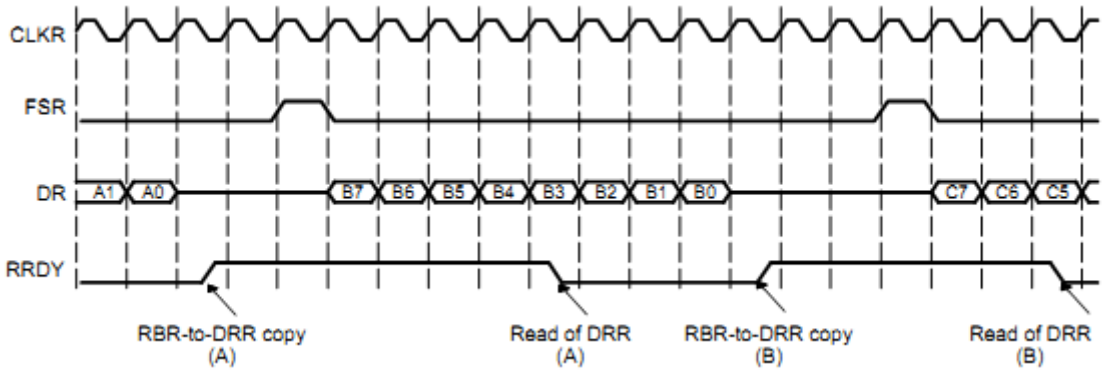
Описание регистра SRGR



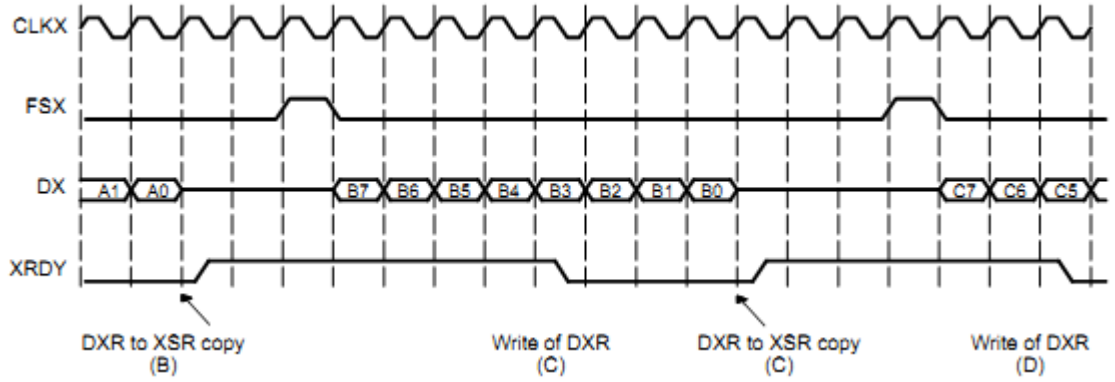
LEGEND: R/W = Read/Write; -n = value after reset

Временные диаграммы операций приема и передачи данных

Прием данных:



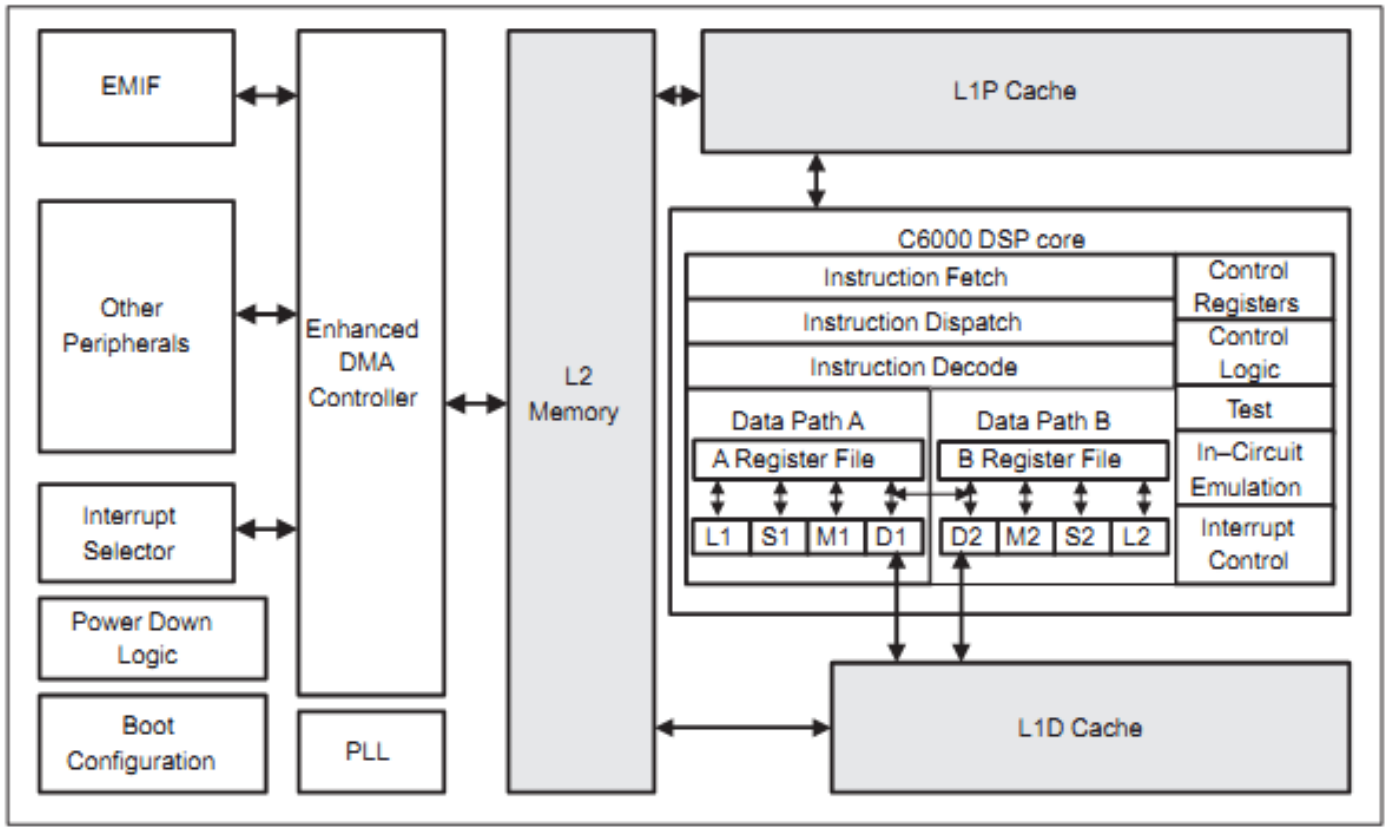
Передача данных:



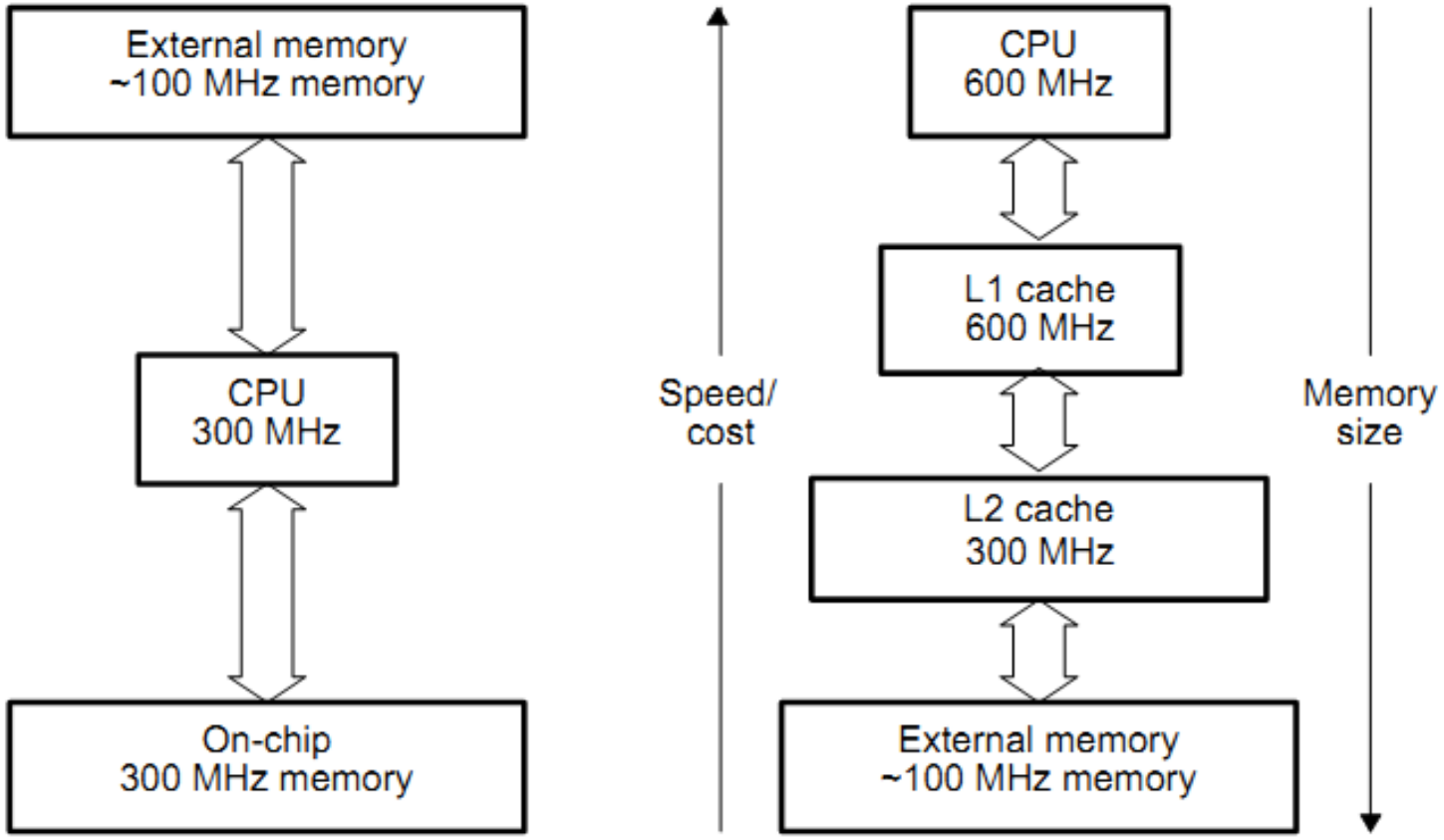
Разработка программного обеспечения для сигнальных процессоров TMS320C64xx

Часть 6. Архитектура памяти.

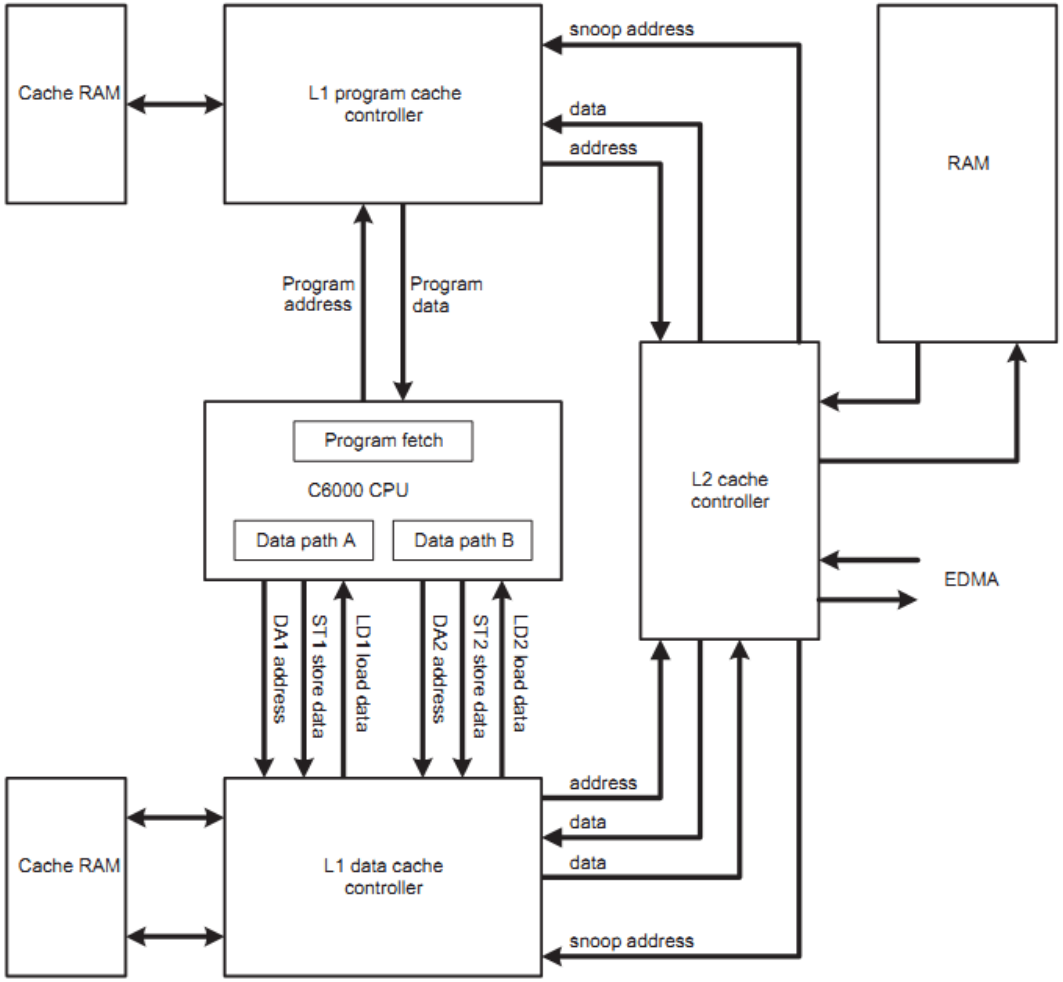
Блок-схема процессоров семействаTMS320C64x



Сравнение одноуровневой и иерархической архитектур памяти

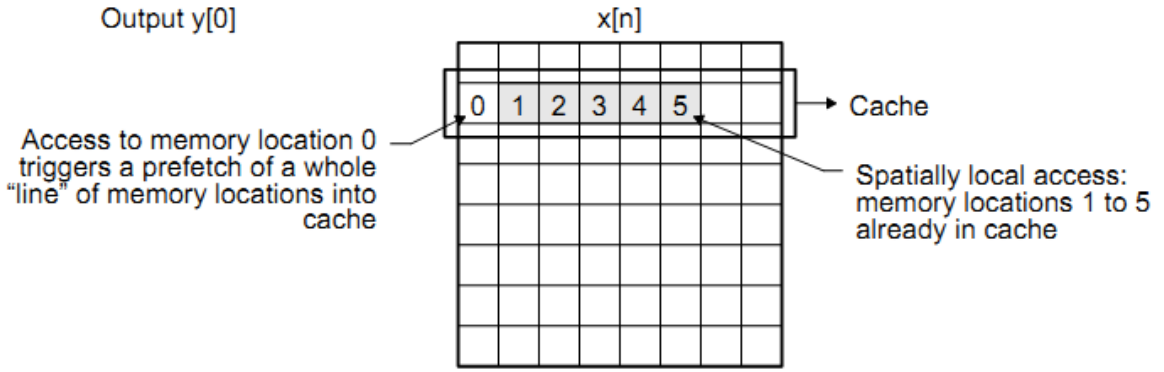


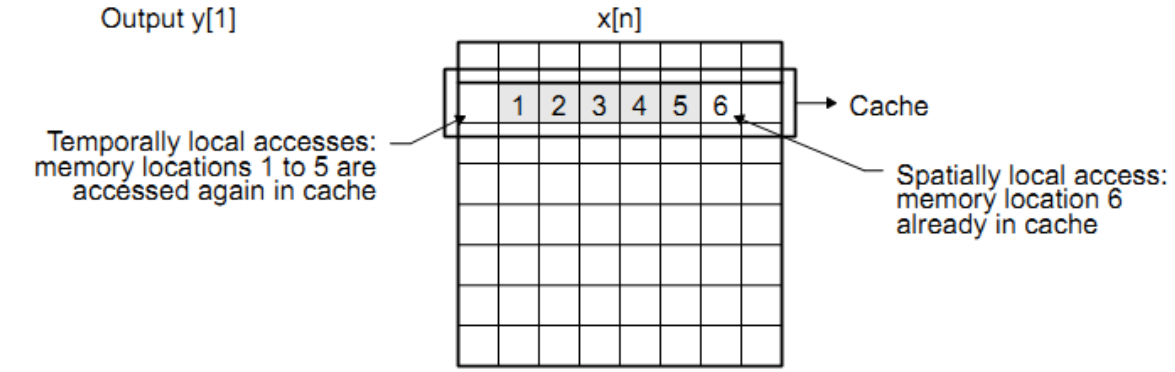
Блок-схема двухуровневой архитектуры памяти процессоров c64x



Принцип локальности на примере FIR фильтра

$$y[0] = h[0] \times x[0] + h[1] \times x[1] + \ldots + h[5] \times x[5]$$
$$y[1] = h[0] \times x[1] + h[1] \times x[2] + \ldots + h[5] \times x[6]$$

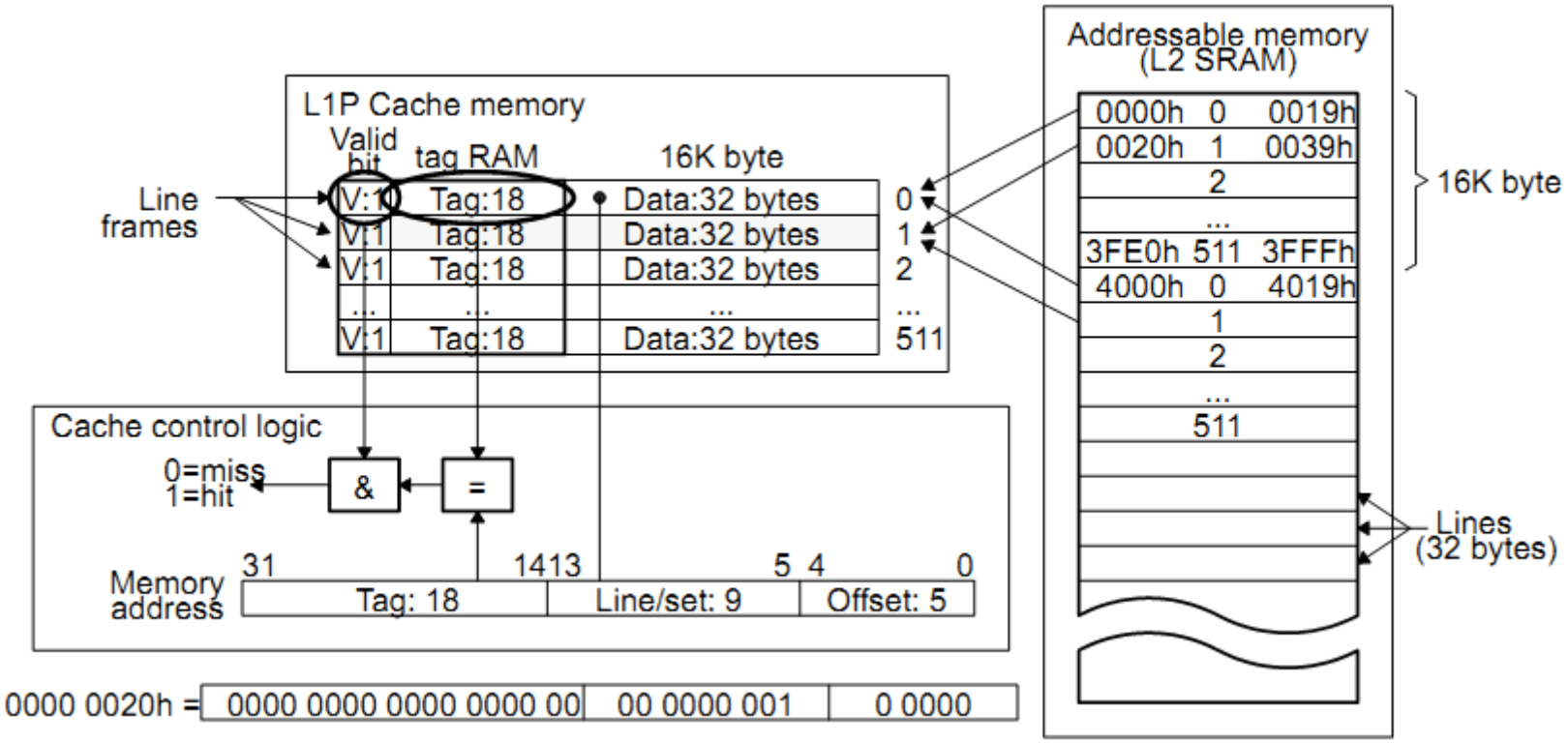




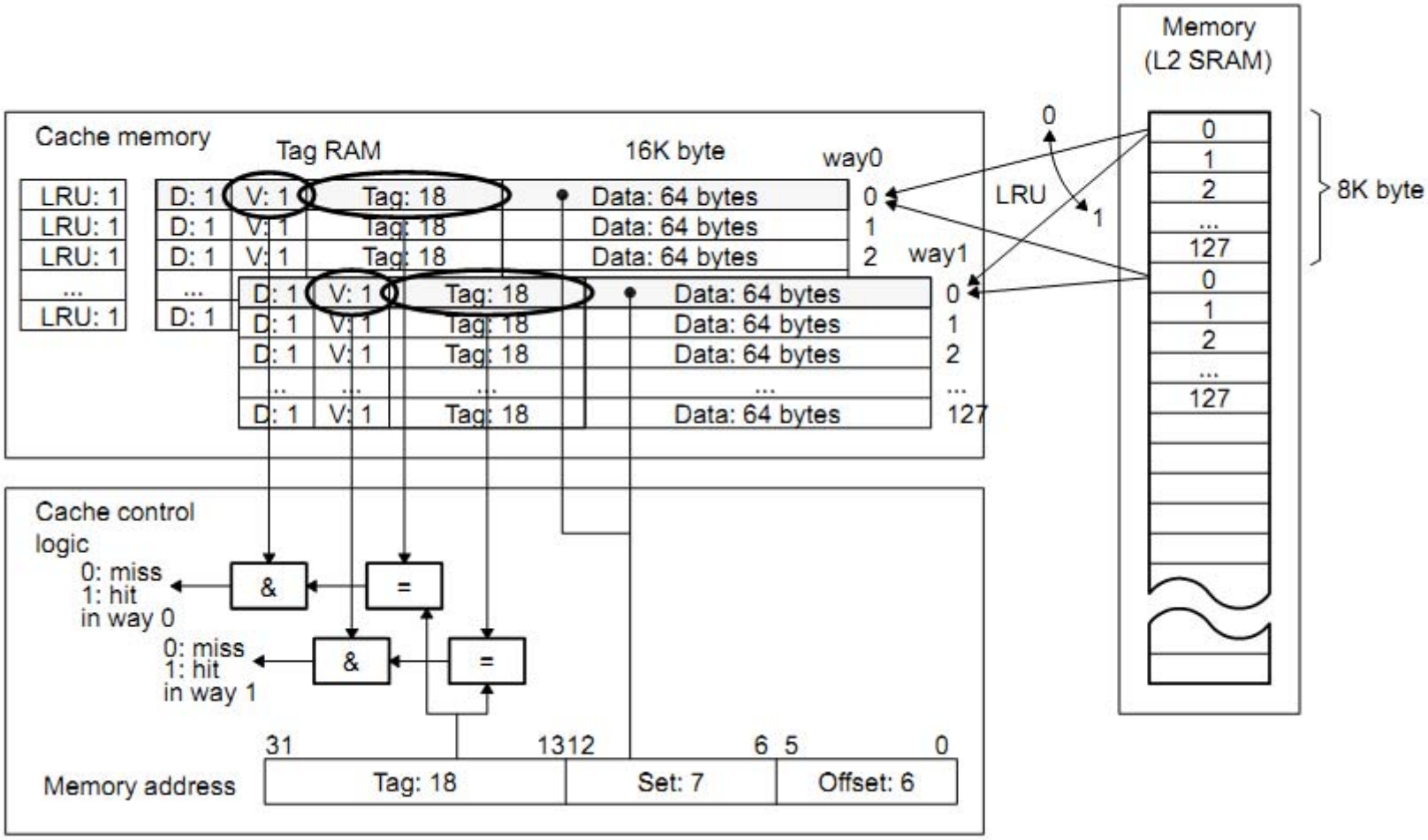
Виды реализаций кэш-памяти

- Кэш-память с прямым отображением;
- Ассоциативная кэш-память;
- Комбинированная реализация.

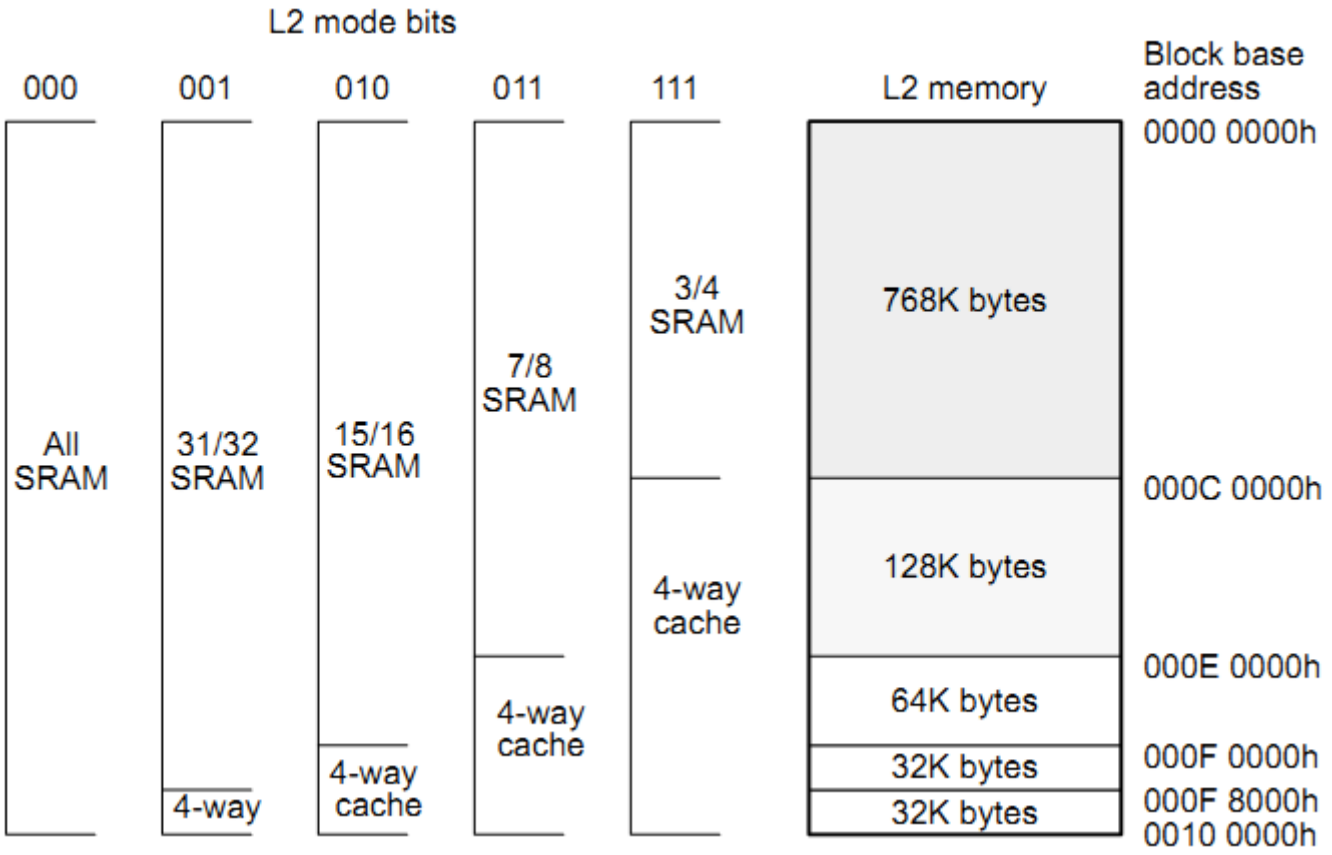
Архитектура кэш-памяти программ



Архитектура кэш-памяти данных



Варианты конфигурации памяти L2



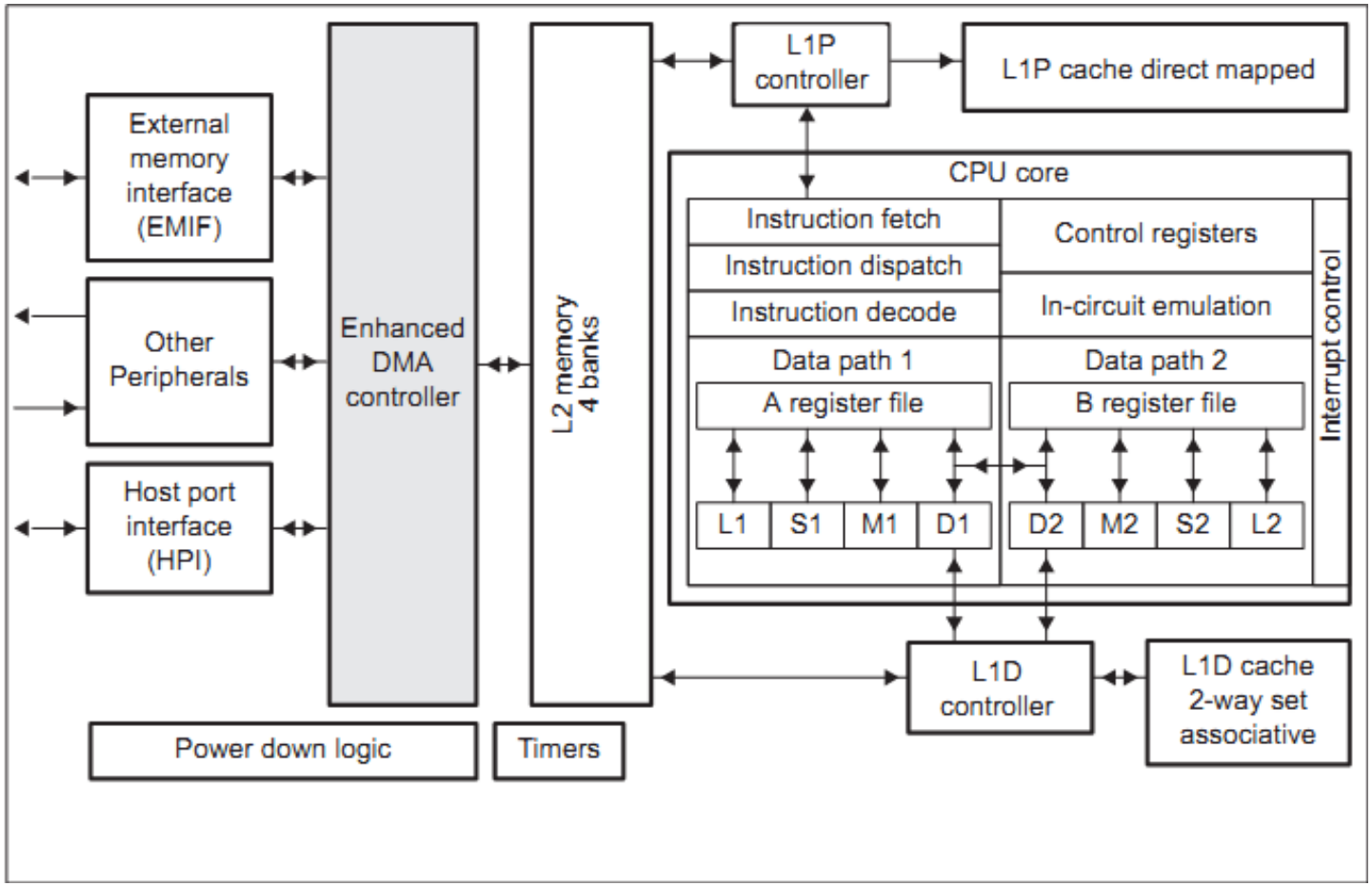
Пример файла линкера с конфигурацией памяти

```
MEMORY
{
       L2SRAM:      origin = 00000000h   length = 000F8000h
       CE0:         origin = 80000000h   length = 01000000h
}

SECTIONS
{
       .cinit       >      L2SRAM
       .text        >      L2SRAM
       .stack       >      L2SRAM
       .bss         >      L2SRAM
       .const       >      L2SRAM
       .data        >      L2SRAM
       .far         >      L2SRAM
       .switch      >      L2SRAM
       .sysmem      >      L2SRAM
       .tables      >      L2SRAM
       .cio         >      L2SRAM
       .external    >      CE0
}
```

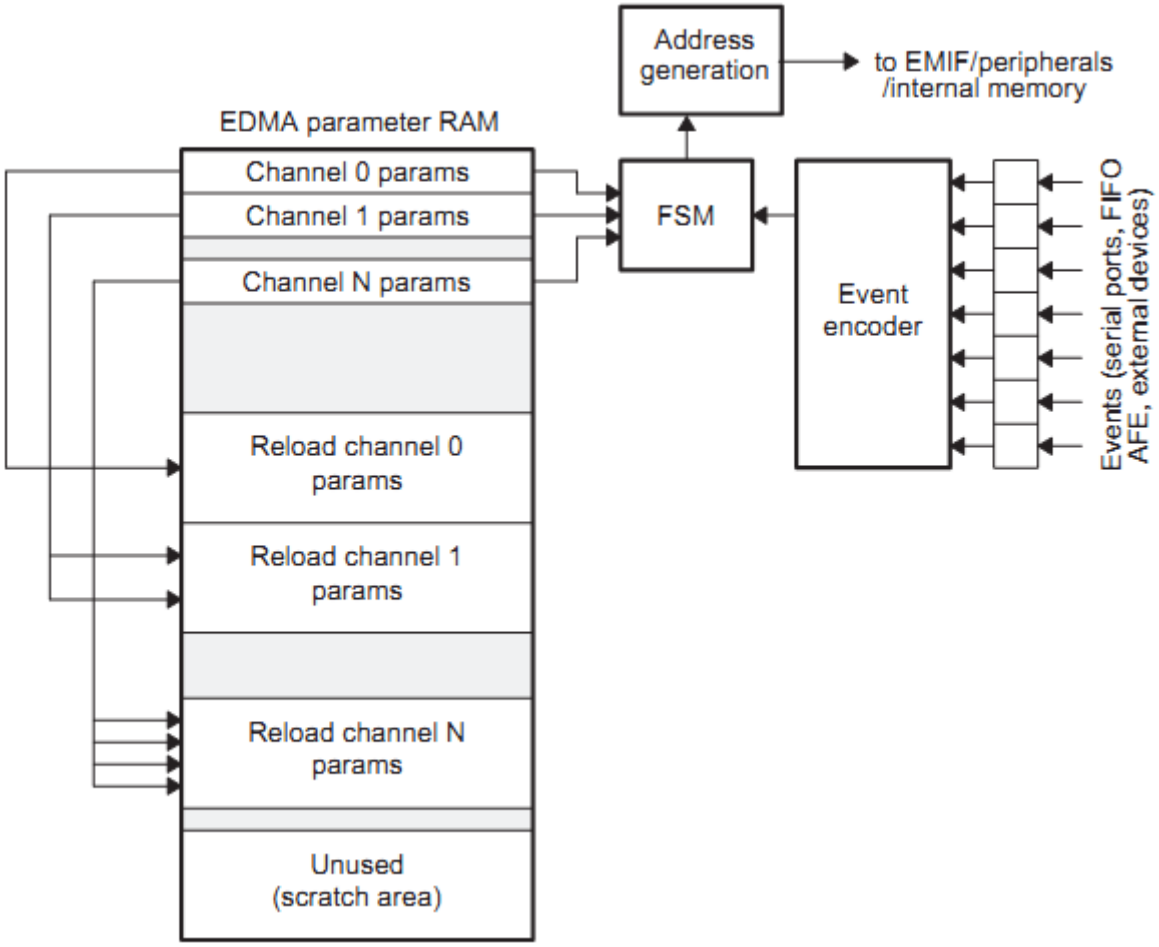
Разработка программного обеспечения для сигнальных процессоров TMS320C64xx

Часть 7. Контроллер EDMA.

Блок-схема процессоров семействаTMS320C64x



Структурная схема контроллера EDMA



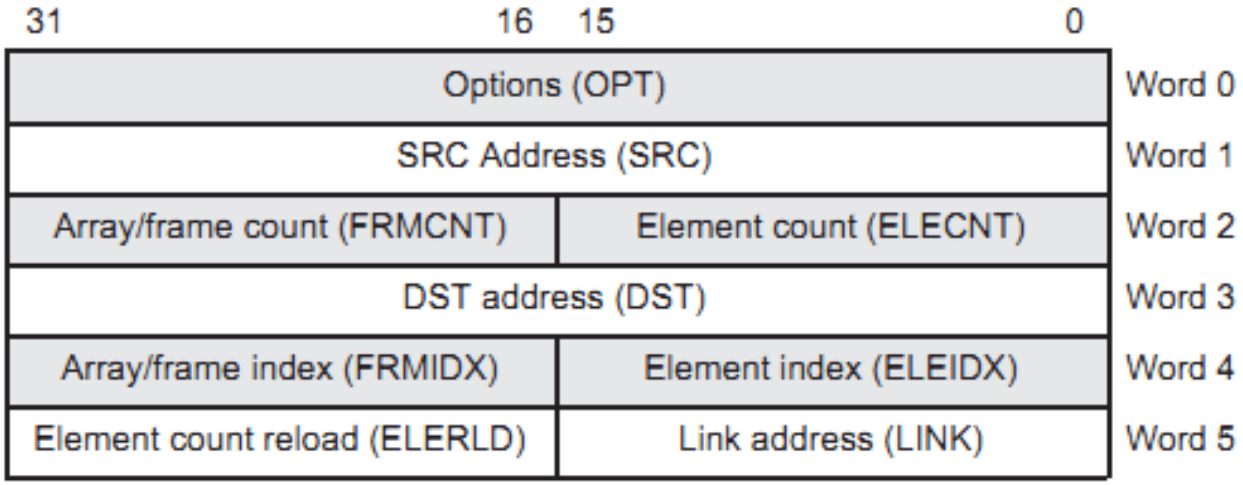
Основные регистры управления EDMA

Регистр	Описание
ER (ERL, ERH)	Event Register – фиксирует факт возникновения события
EER (EERL, EERH)	Event Enable Register – разрешение/запрещение обработки определено события (-ий) контролером
ECR (ECRL, ECRH)	Event Clear Register - запись в него сбрасывает биты в ER
ESR (ESRL, ESRH)	Event Set Register - запись в него устанавливает биты в ER
EPR (EPRL, EPRH)	Event Polarity Register – задает полярность для сигнала события
CIER (CIERL, CIERH)	Channel Interrupt Enable Register – разрешает/запрещает генерацию прерывания CPU для заданного канала
CIPR (CIPRL, CIPRH)	Channel Interrupt Pending Register – содержит информацию о том какой канал вызвал прерывание CPU

Список событий синхронизации для контроллера EDMA процессоров C64x

EDMA Channel Number	Event Acronym	Event Description
0	DSPINT	Host-to-DSP interrupt
1	TINT0	Timer 0 interrupt
2	TINT1	Timer 1 interrupt
3	SD_INTA	EMIFA SDRAM timer interrupt
4	GPINT4/EXT_INT4	GPIO event 4/External interrupt 4
5	GPINT5/EXT_INT5	GPIO event 5/External interrupt 5
6	GPINT6/EXT_INT6	GPIO event 6/External interrupt 6
7	GPINT7/EXT_INT7	GPIO event 7/External interrupt 7
8	GPINT0	GPIO event 0
9	GPINT1	GPIO event 1
10	GPINT2	GPIO event 2
11	GPINT3	GPIO event 3
12	XEVT0	McBSP0 transmit event
13	REVT0	McBSP0 receive event
14	XEVT1	McBSP1 transmit event
15	REVT1	McBSP1 receive event
16	–	None
17	XEVT2	McBSP2 transmit event
18	REVT2	MCBSP2 receive event
19	TINT2	Timer 2 interrupt
20	SD_INTB	EMIFB SDRAM timer interrupt
21	PCI	PCI Wakeup Interrupt
22	–	None
23	–	None
24	–	None

Список параметров конфигурации канала EDMA

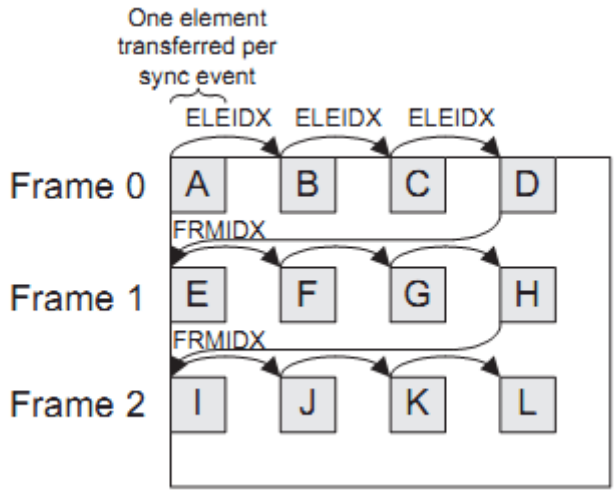


Список полей регистра OPT

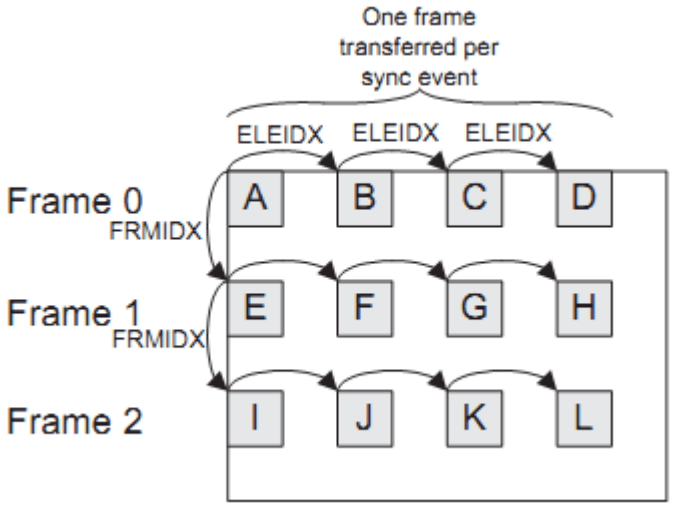
Бит №	Поле	Описание
31-29	PRI	Уровни (4) приоритета для событий EDMA
28-27	ESIZE	Размер элемента (32, 16 и 8 бит)
26	2DS	Размерность источника данных (1-D, 2-D)
25-24	SUM	Режим обновления адреса источника (фиксированный, инкремент, декремент, шаг заданный в регистрах ELEIDX и FRMIDX)
23	2DD	Размерность получателя данных (1-D, 2-D)
22-21	DUM	Режим обновления адреса получателя (фиксированный, инкремент, декремент, шаг заданный в регистрах ELEIDX и FRMIDX)
20	TCINT	Разрешает генерацию прерывания по завершении передачи
19-16	TCC	4-х битный код используемый для выставления бита прерывания в регистре CIPR
1	LINK	Связывание параметров конфигурации разрешено
0	FS	Использовать кадровую синхронизацию

Типы пересылок EDMA: 1-D transfer

Пересылка с поэлементной синхронизацией:

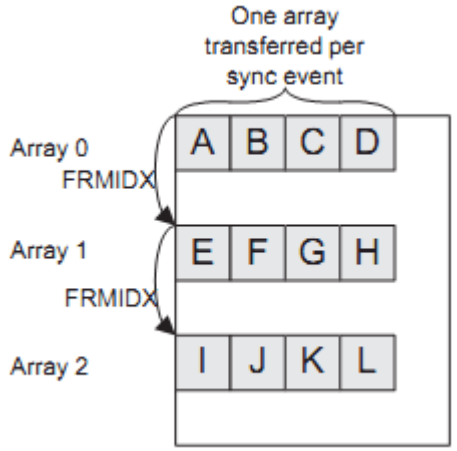


Пересылка с покадровой синхронизацией:

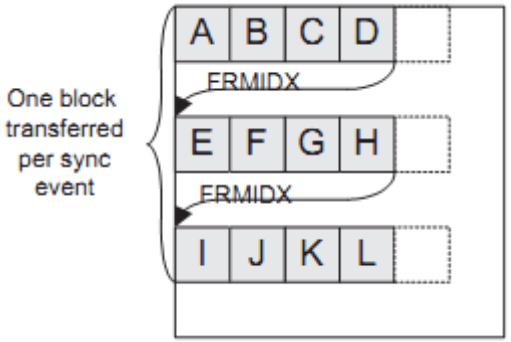


Типы пересылок EDMA: 2-D transfer

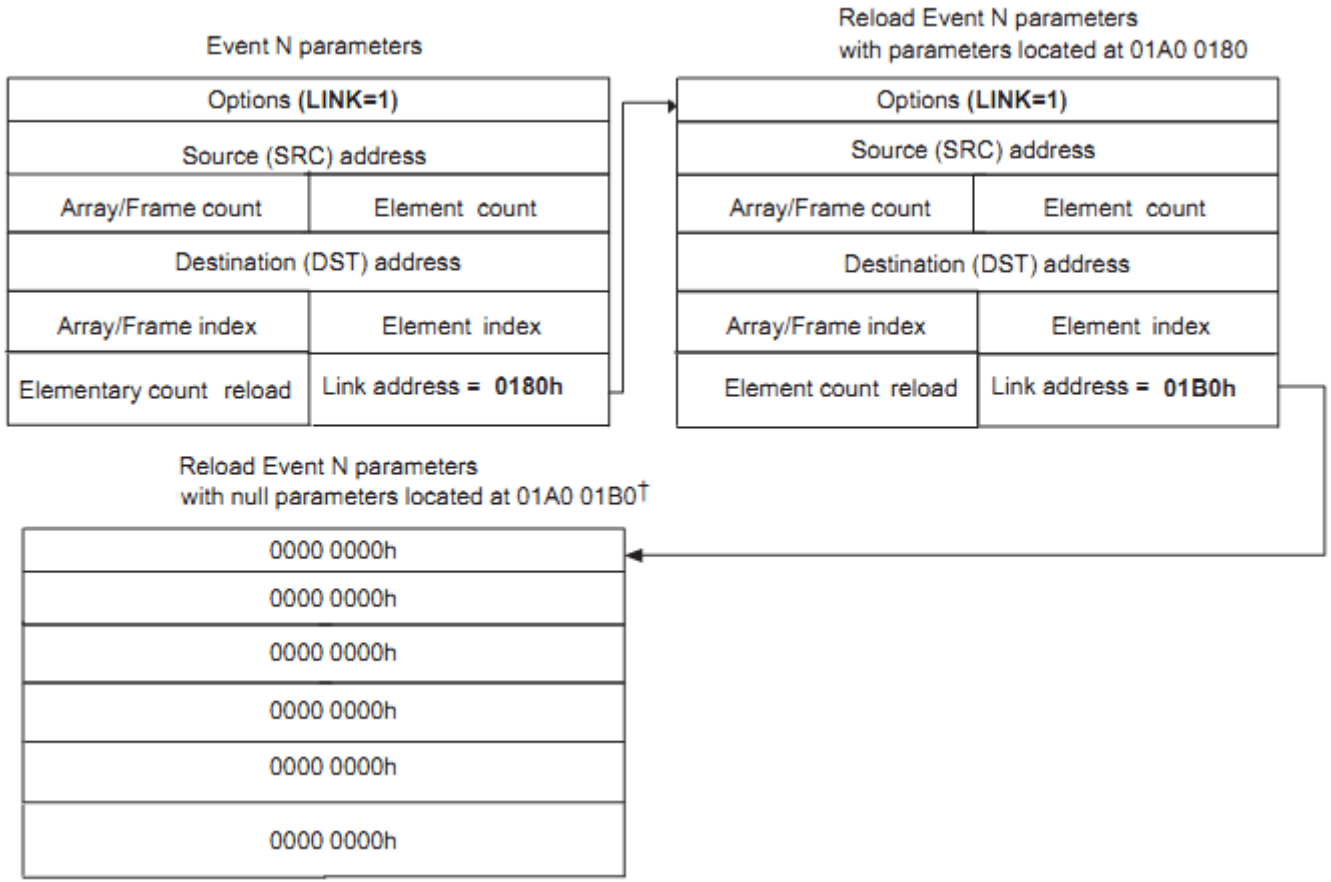
Пересылка по событию синхронизации на каждый массив:



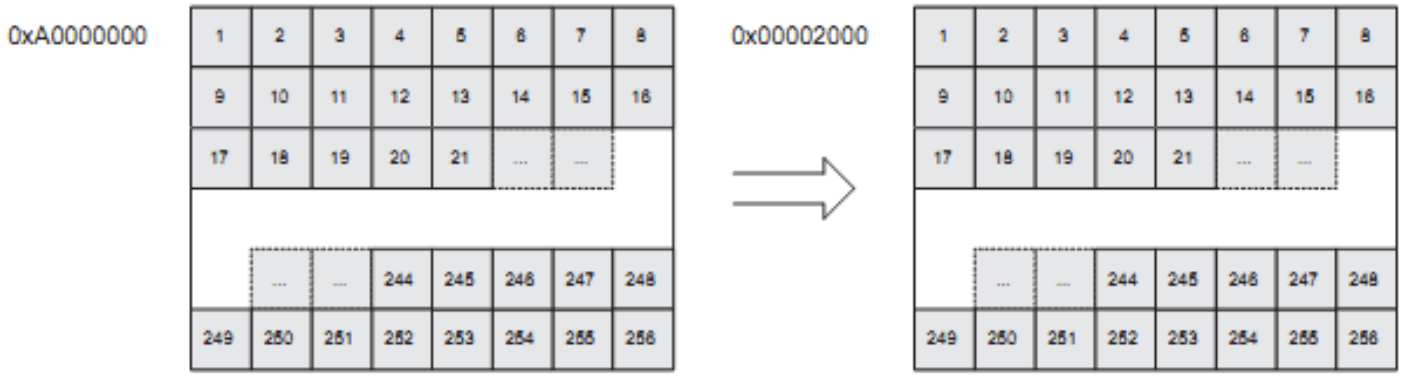
Пересылка с поблочной синхронизацией:

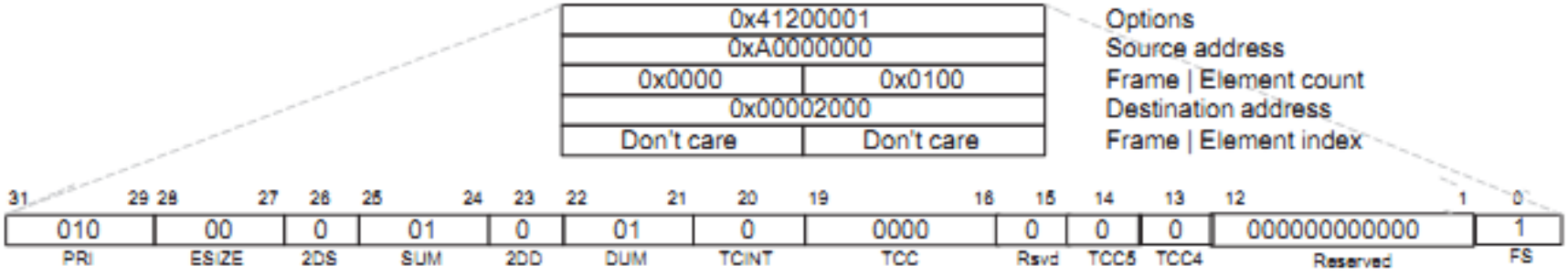


Связывание пересылок

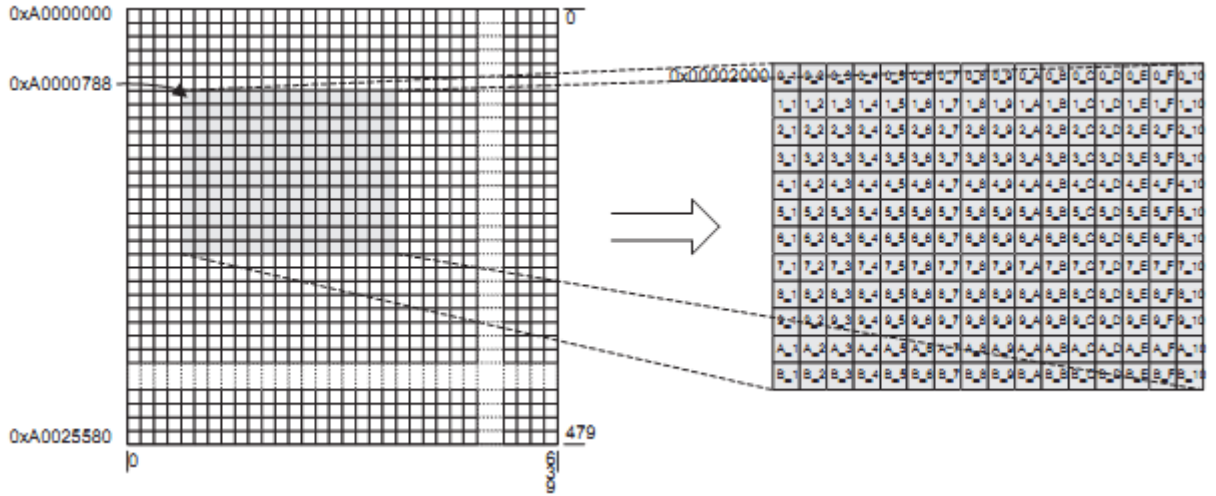


Примеры EDMA транзакций: перемещение блока данных





Примеры EDMA транзакций: извлечение субэлемента блока данных

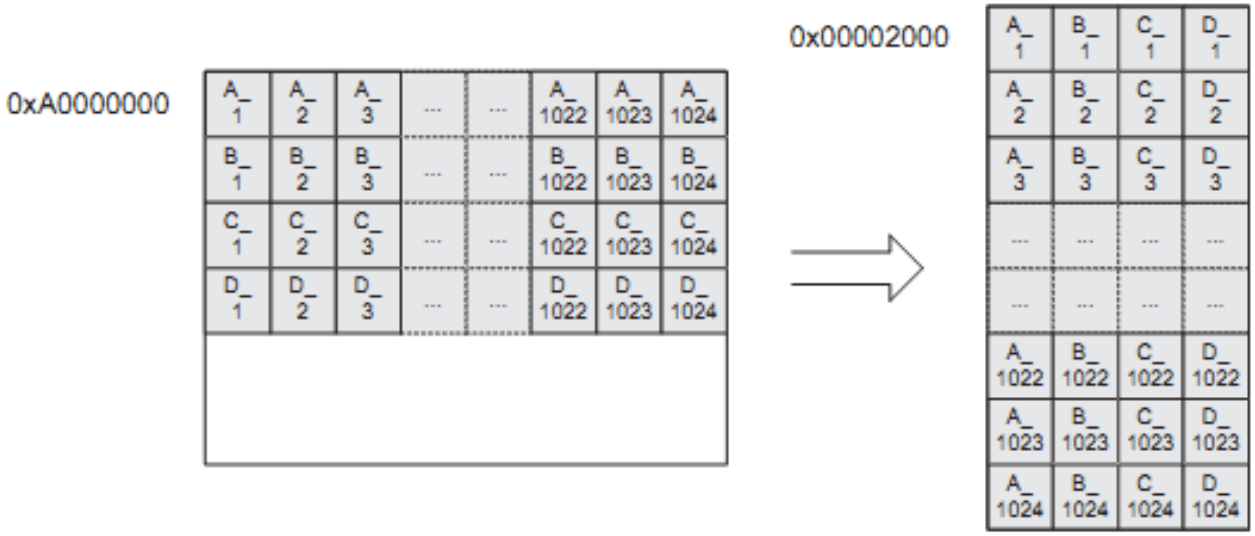


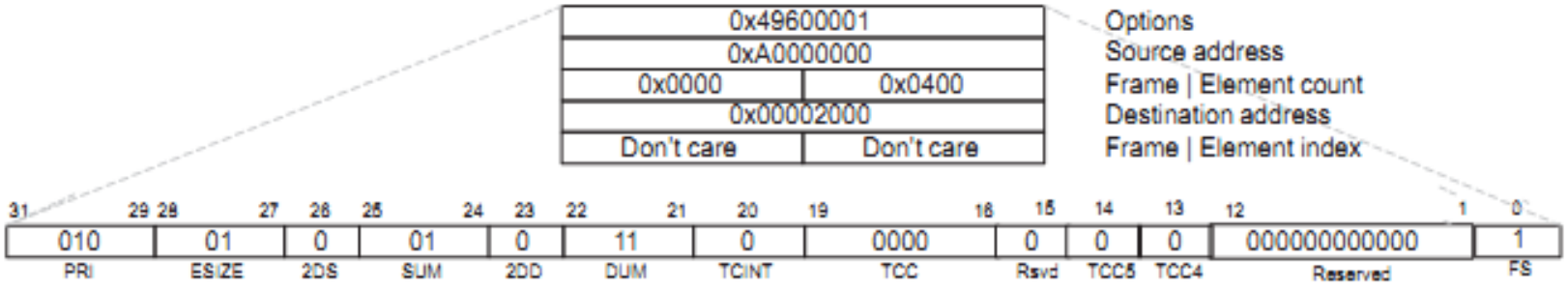
0x4D200001	
0xA0000788	
0x000B	0x0010
0x00002000	
Don't care	Don't care

Options
Source address
Frame | Element count
Destination address
Frame | Element index

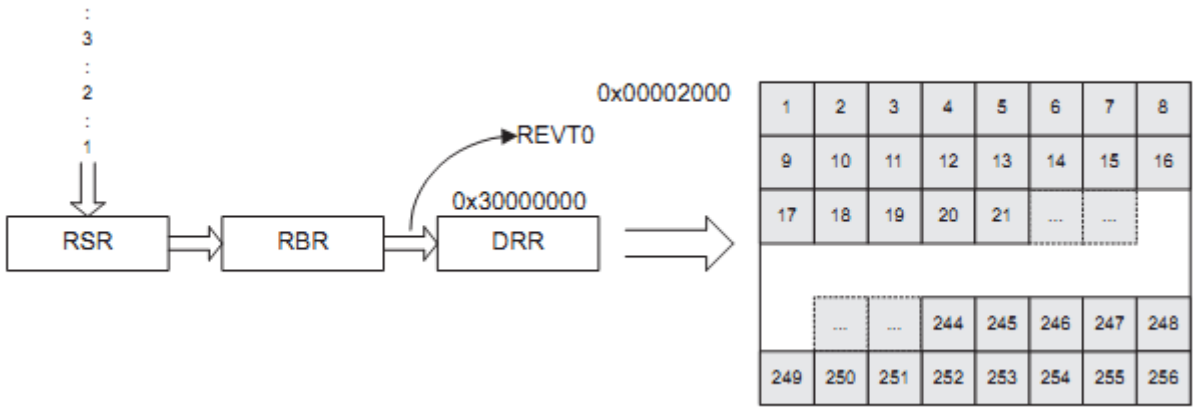
31 29	28 27	26	25 24	23	22 21	20	19 16	15	14	13	12 1	0
010	01	1	01	0	01	0	0000	0	0	0	000000000000	1
PRI	ESIZE	2DS	SUM	2DD	DUM	TCINT	TCC	Rsvd	TCC5	TCC4	Reserved	FS

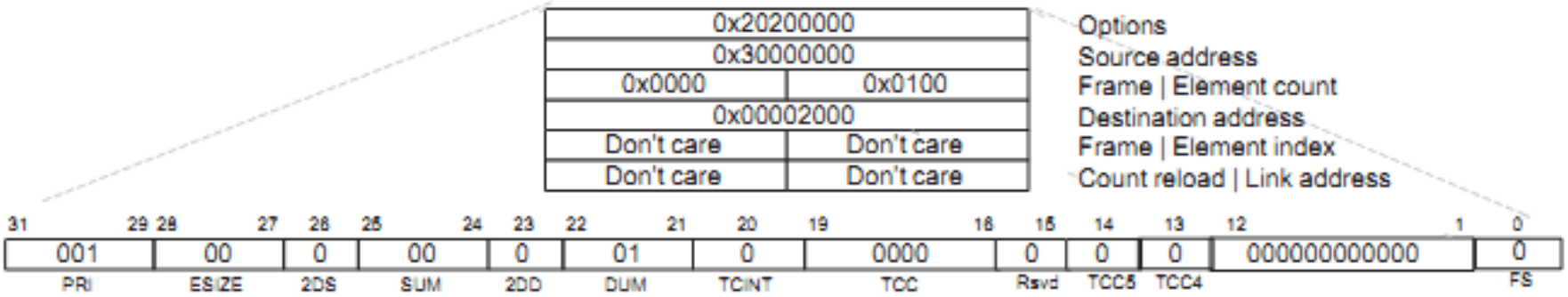
Примеры EDMA транзакций: перестановка данных



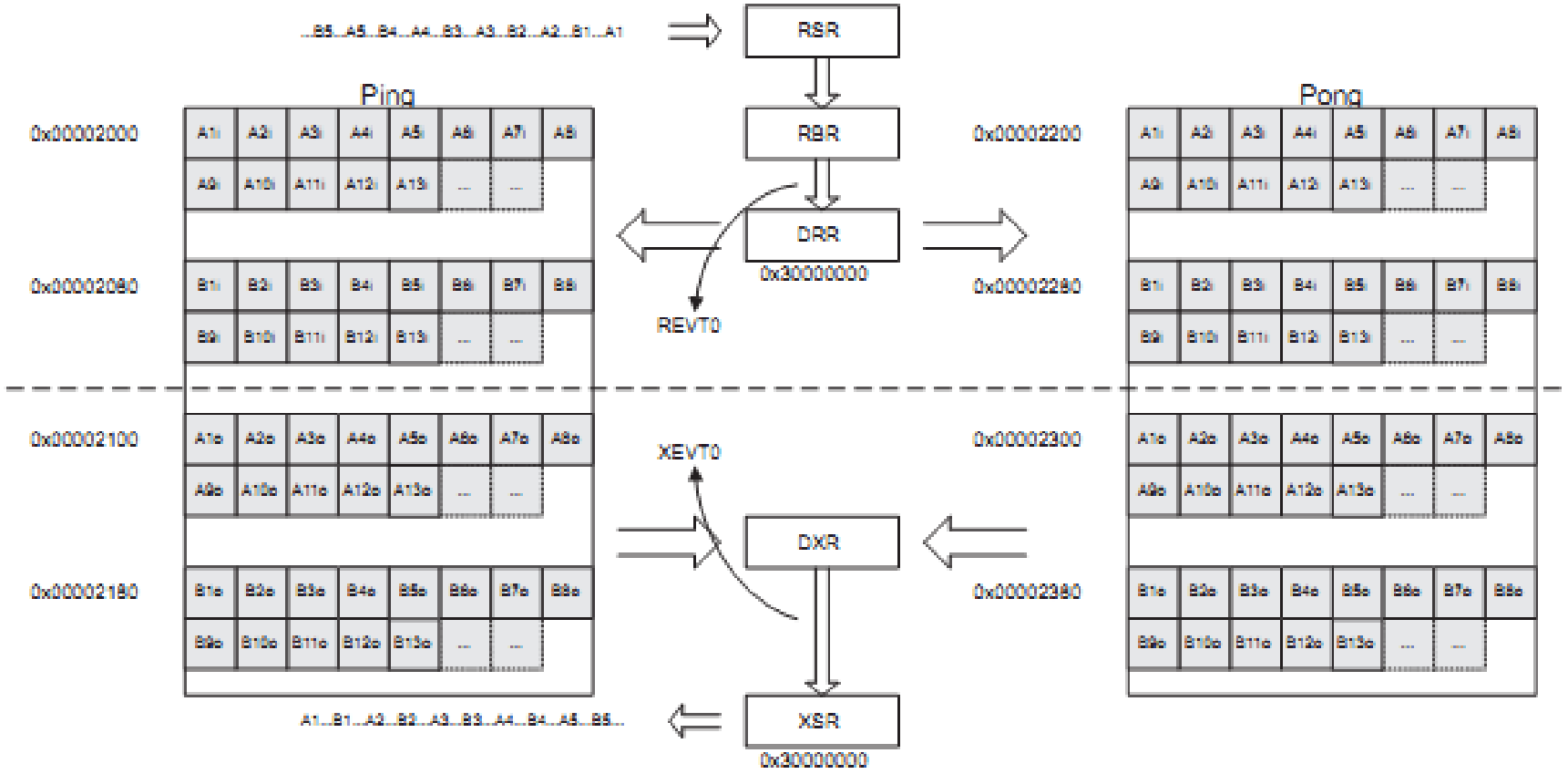


Примеры EDMA транзакций: чтение данных из McBSP

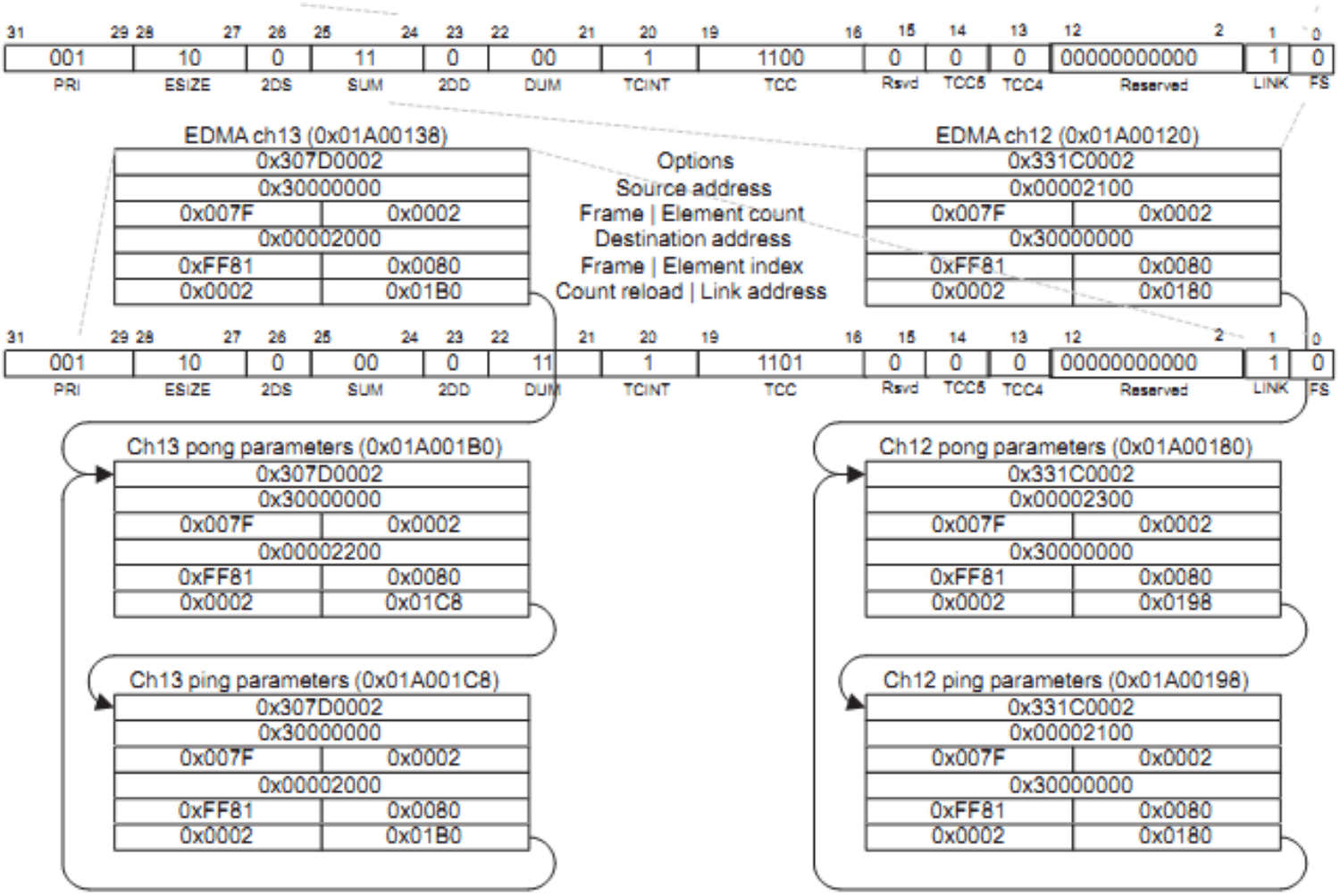




Примеры EDMA транзакций: работа с McBSP- ping-pong buffering (схема)



Примеры EDMA транзакций: работа с McBSP - ping-pong buffering (конфигурация)



Разработка программного обеспечения для сигнальных процессоров TMS320C64xx в IDE Code Composer Studio

Часть 8. Основные возможности среды разработки CCS.

Процесс разработки ПО для систем ЦОС на базе ЦСП

Разработка и отладка программного кода на языке С или ассемблере в соответствии с выбранным алгоритмом. **Результат** – набор текстовых файлов с программным кодом.

Оптимизация программного кода с учетом архитектуры выбранного ЦСП. **Результат** – один бинарный файл, содержащий цифровое двоичное представление разработанного программного кода.

Тестирование и отладка разработанного программного кода на симуляторе ЦСП.

Тестирование и отладка программного кода на целевой плате.

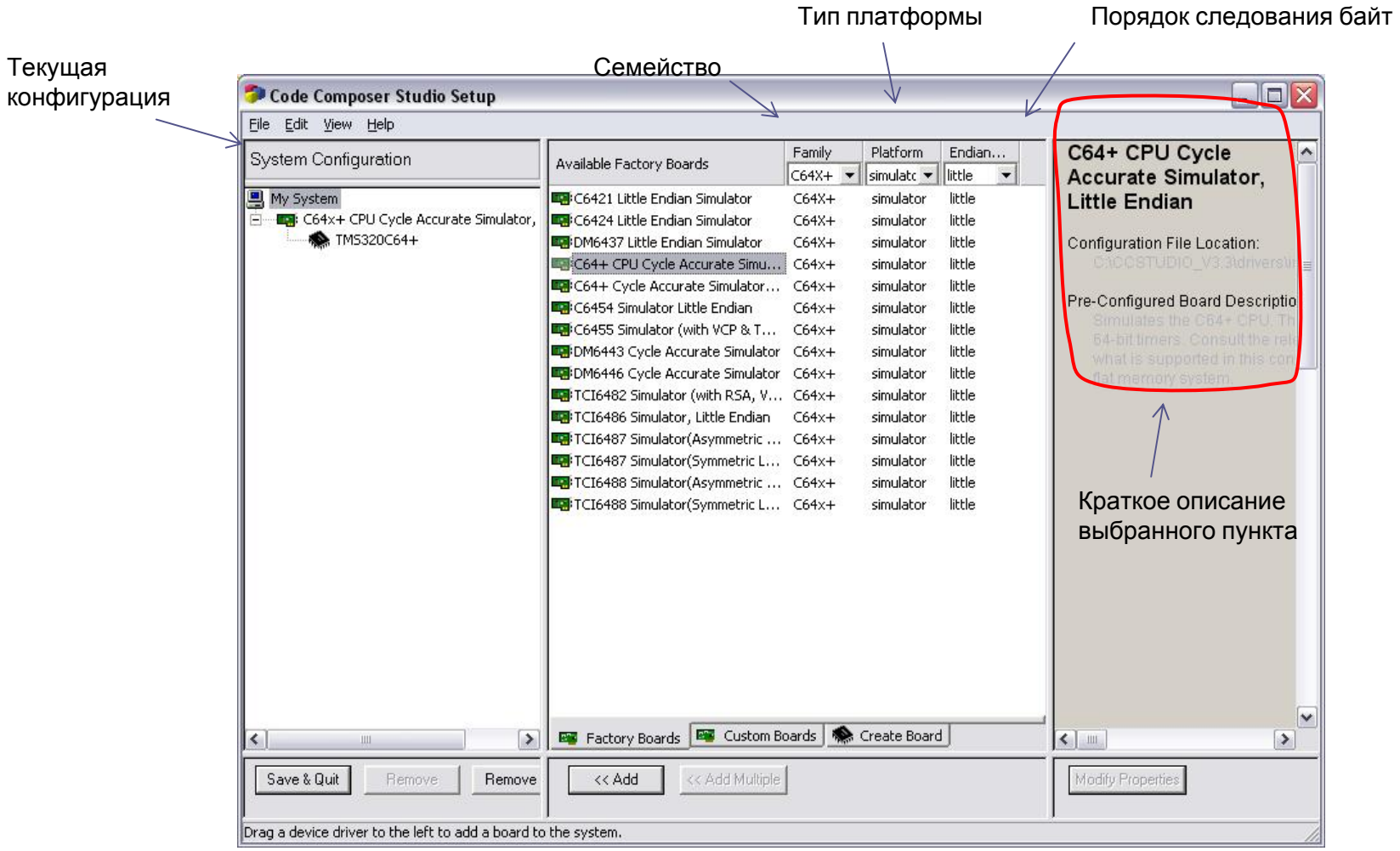
Средства разработки и отладки ПО фирмы Texas Instruments для ЦСП семейства C6000

- Единый, интегрированный в каждый ЦСП фирмы TI, отладочный интерфейс JTAG, позволяющий подключить несколько ЦСП к одному отладчику всего по 6 проводам и обеспечивающий высокую скорость обмена данными;
- Технологию обмена данными в реальном времени – RTDX, базирующуюся на интерфейсе JTAG;
- Среду разработки ПО Code Composer Studio.

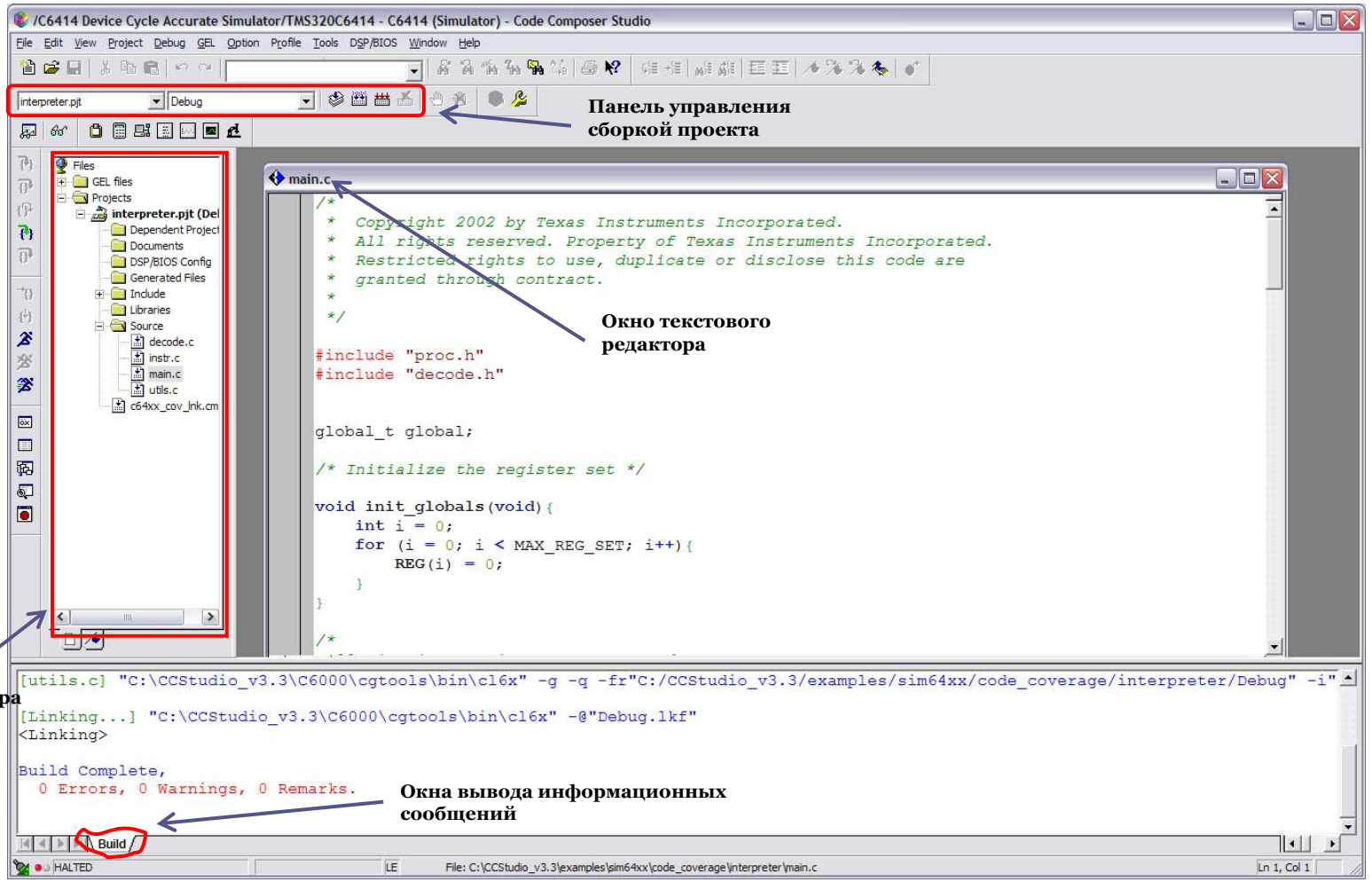
IDE Code Composer Studio включает в себя:

- Интегрированную среду разработки, состоящую из редактора, отладчика, менеджера проектов;
- C/C++ компилятор, оптимизатор ассемблерного кода, компоновщик;
- Симулятор ЦСП;
- Загрузчик программного кода в ЦСП;
- Мультипроцессорный отладчик, оптимизированный для приложений ЦОС, который содержит набор средств анализа и отладки программ в реальном времени, базирующийся на RTDX (механизм обмена данными между хостом и целевым устройством в реальном времени).

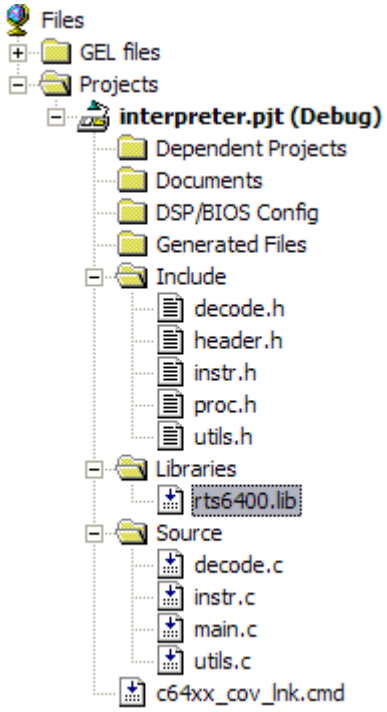
Утилита Setup среды CCS



Основные элементы интерфейса программы Code Composer Studio



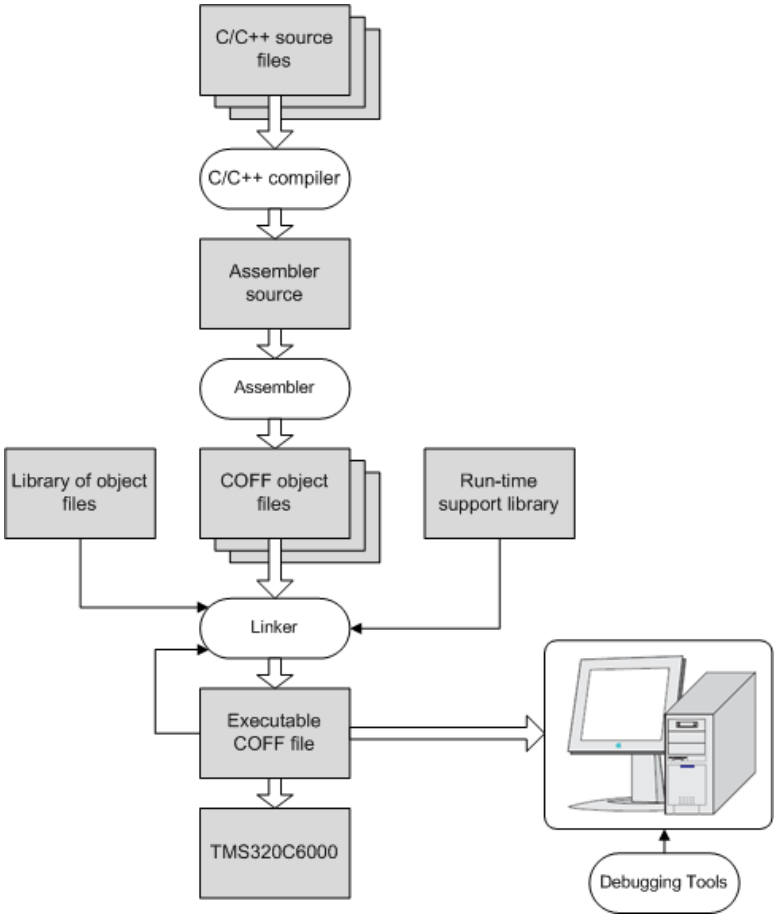
Структура проекта CCS



Проект CCS состоит из следующих элементов:

- Папка ***Documents*** содержит файлы не используемые при компиляции проекта;
- Папка ***DSP/BIOS Config*** содержит файлы конфигурации DSP/BIOS;
- Папка ***Generated Files*** содержит временные файлы создаваемые в процессе компиляции;
- Папка ***Include*** содержит заголовочные файлы, в которых объявлены константы, определения функций и глобальных переменных (расширение .h);
- Папка ***Libraries*** содержат файлы библиотек, представляющих собой реализации различных, часто употребляемых функций (расширение .lib);
- Папка ***Source*** содержит файлы исходных кодов программы (расширение .asm и .c);
- Файлы компоновщика (расширение .cmd).

Процесс сборки проекта



Окна Code Composer Studio в режиме отладки