Contents

Processor architecture

X16 and X16E cores

North MpMII

5

West MpMII

Routing Engine

East MpMII

South MpMII

4

4096 Transaction cache

Frame Processing System

TXCR15

TXCR0

6

Stream controller

4 X FIFO

ARAM 256

4 descriptors

7

T M U X

9

Cache subsystem 64 Kb

SDRAM ISI

10

FFT Machine

AR data

R data

Control

1

AFR15

AFR0

R15

R0

Integer/FP ALU

AR15

AR0

ATU

Sequencer

Prefetcher & Code cache 32Kb

2

Context controller

Memory management RAM 256 entries

Peripheral devices ISI

8

External SPI interface

Bootloader ROM & SRAM 32Kb
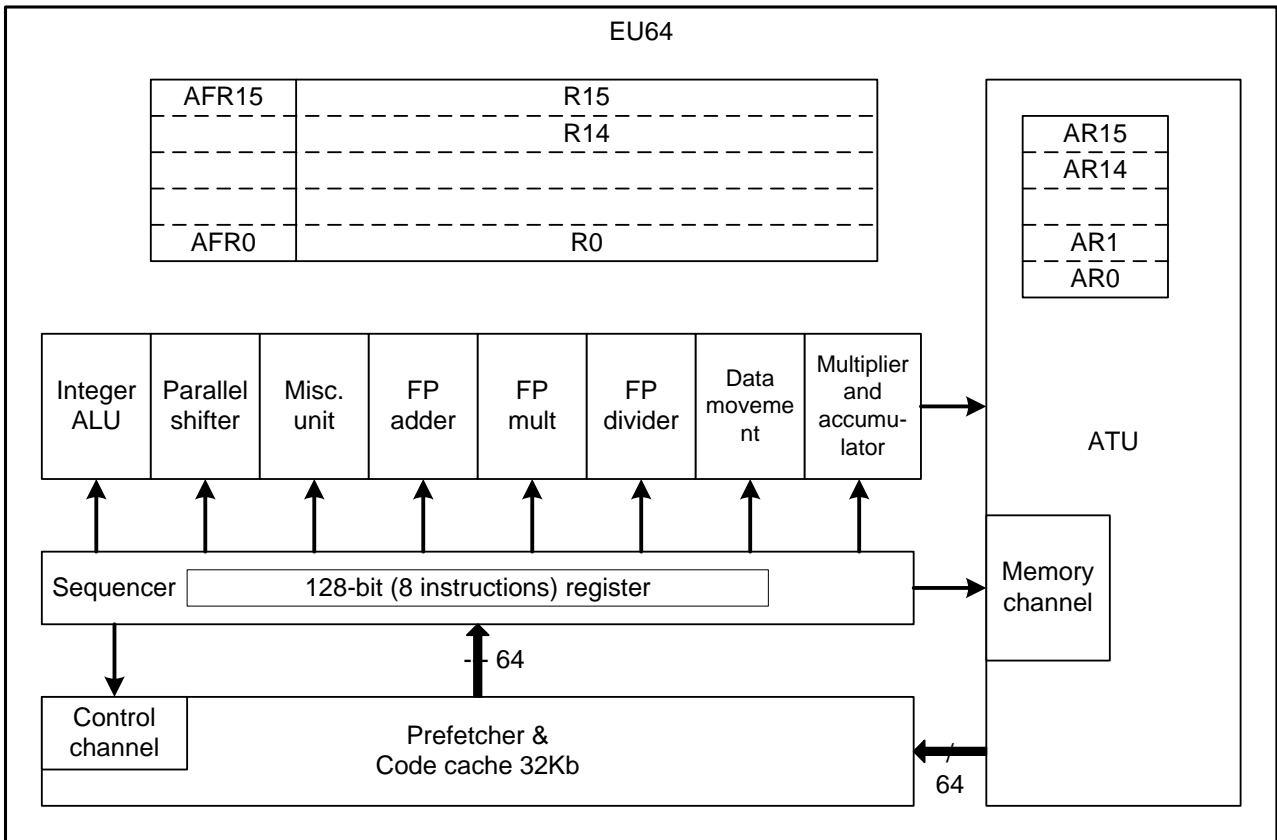
SPI

3

Messenger

1. The executive unit, capable of running up to 8 instructions at a time. The EU consists of:
   - Arithmetic logic unit containing 16 128-bit general-purpose registers - R[15: 0] and identified with them 16 32-bit flags registers AFR[15: 0];
   - Address translation unit containing 16 address registers AR[15: 0];
   - Sequencer, which manages the process of sending instructions to execution as their operands and receivers of transaction results are ready;
   - Prefetcher, which forms a instruction flow to the sequencer and controls the processes of control transfer within one code object, which is performed by the instructions of the loop, conditional and unconditional jumps, calling subroutines and returning from subroutines;
2. A context controller that manages processes to store the core context in the PSO (Process State Object) and load the context from the PSO. The context controller also contains a buffer cache for 128 positions of free memory block selectors and 128 selector positions for free entry points in the descriptor table. This buffer, together with the firmware, allows the context controller to process high-level instructions for allocating and deleting blocks of memory.
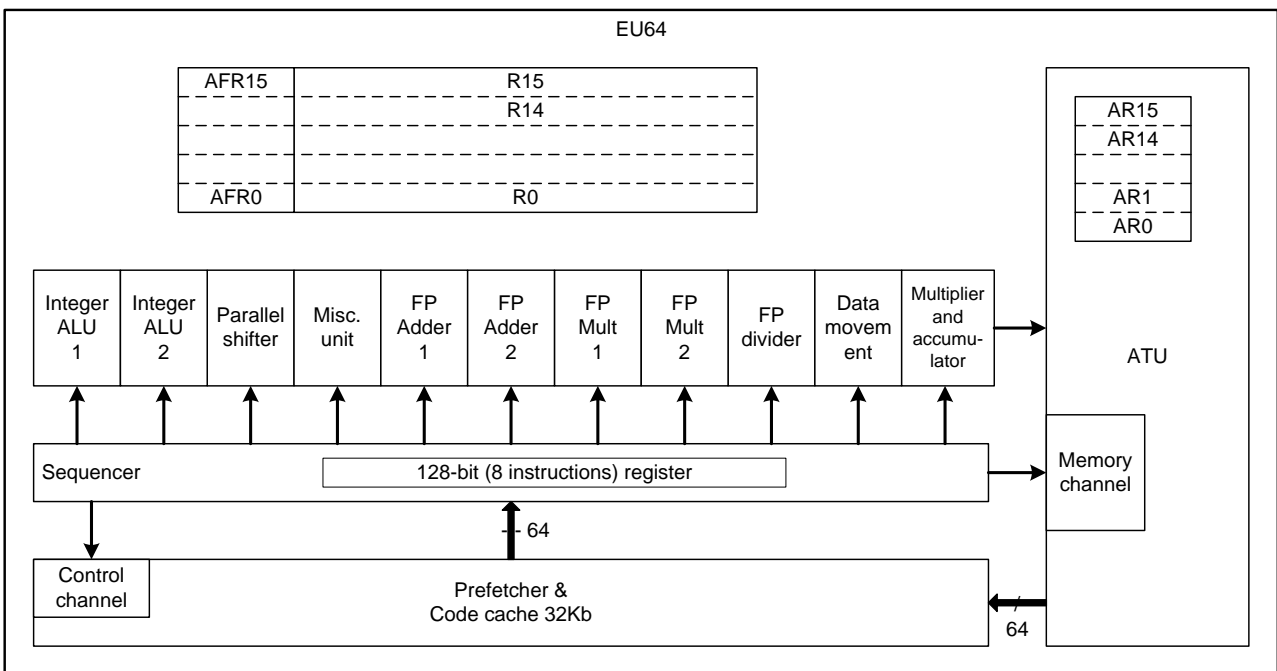
3. The messenger is the system responsible for setting messages in the process message queue and for immediately calling procedures for handling exceptions, hardware interrupts, and debugging interrupts.

4. The Frame Processing Unit (FPU) is the unit responsible for forming frames for the multiprocessor network, receiving frames from the network, disassembling them, generating read / write transactions for local RAM, sending messages to Messenger, and sending responses to message frames and frames of reading data from memory. The unit contains a cache of transaction descriptors with a capacity of 4096 cells, serving the caching of 16 transactions of each of the 255 network processors. The TXCR [15: 0] registers (MasterRAM location) contain the descriptors of the multiprocessor transactions generated by the processor core.

5. Routing Engine - 5-port switch multiprocessor frames. RTE carries out routing of transit frames, outgoing frames and selection of incoming frames. It has 4 32-bit MpMII interfaces (Multiprocessor Media Independent Interface) north, east, south and west. MpMII can connect to blocks that implement the physical layer of the interprocessor connection, can connect to the RTE of another core, if several cores are combined on the same chip in a cluster.

6. Stream Controller. It implements the mechanism of streaming data exchange between processes both within a single core and in a multiprocessor network. Stream controller contains a file of registers for 256 thread descriptors, as well as a cache for 4 threads, which can be used by the core at the same time.

7. TMUX – transaction multiplexer. TMUX processes requests for access to local memory regions from internal processor modules. TMUX defines the region of memory that is accessed and sends the transaction for execution in the corresponding local device. 2 ISI (InSystem Interface) interfaces are provided by TMUX for connection of SDRAM controllers and various peripheral devices.

8. An SPI interface unit with a built-in memory buffer containing the startup code and a small memory buffer that is required to perform the initial system initialization procedure. The memory buffer is also used to form packet data transfers between the processor and an external device connected via SPI.

9. Cache subsystem. Used to cache data read from SDRAM, data buffering, written in SDRAM. The block contains a 4-way associative cache and four 32-byte write buffers. Data exchange between Cache subsystem and SDRAM is performed by blocks of 32 bytes.

10. FFT Machine. The machine calculates the FFT using complex numbers, the real and imaginary parts of which are represented in a single-precision floating-point format. The maximum length of the data block is 2^{20} values.

Differences between X16 and X16E

The differences between the X16 and X16E cores are in the structure of the execution units of the processor. EU X16 contains a 10-channel sequencer that allows you to run one instruction for an integer ALU, for a floating-point adder and for a floating point multiplier simultaneously.

EU64

AFR15	R15
	R14
AFR0	R0

AR15
AR14
AR1
AR0

Integer ALU	Parallel shifter	Misc. unit	FP adder	FP mult	FP divider	Data movement	Multiplier and accumu-lator

ATU

Sequencer | 128-bit (8 instructions) register

Memory channel

← 64

Control channel | Prefetcher & Code cache 32Kb

/ 64

EU X16E contains a 13-channel sequencer that allows you to simultaneously run 2 instructions for integer operations, 2 add / subtract instructions with floating point and 2 floating point multiplication instructions.

EU64

AFR15	R15
	R14
AFR0	R0

AR15
AR14
AR1
AR0

Integer ALU 1	Integer ALU 2	Parallel shifter	Misc. unit	FP Adder 1	FP Adder 2	FP Mult 1	FP Mult 2	FP divider	Data movement	Multiplier and accumu-lator

ATU

Sequencer | 128-bit (8 instructions) register

Memory channel

← 64

Control channel | Prefetcher & Code cache 32Kb

/ 64

FlyBy Instructions

FlyBy is a class of instructions that allow the processor to execute all subsequent instructions located in the program after the FlyBy instruction, without

waiting for the FlyBy instruction to complete. FlyBy instructions are executed using a hardware resource separate from the execution unit.

A FlyBy instruction that is started for processing by one process can continue processing data even when the EU is switched to processing another process.

FlyBy-instruction may not start processing data if there are no free hardware resources to do this. The operation of the instruction can be checked by analyzing the ZF flag of one of the AFRs, in which the 1-bit status of presence (1) / absence (0) of a free hardware resource is placed.
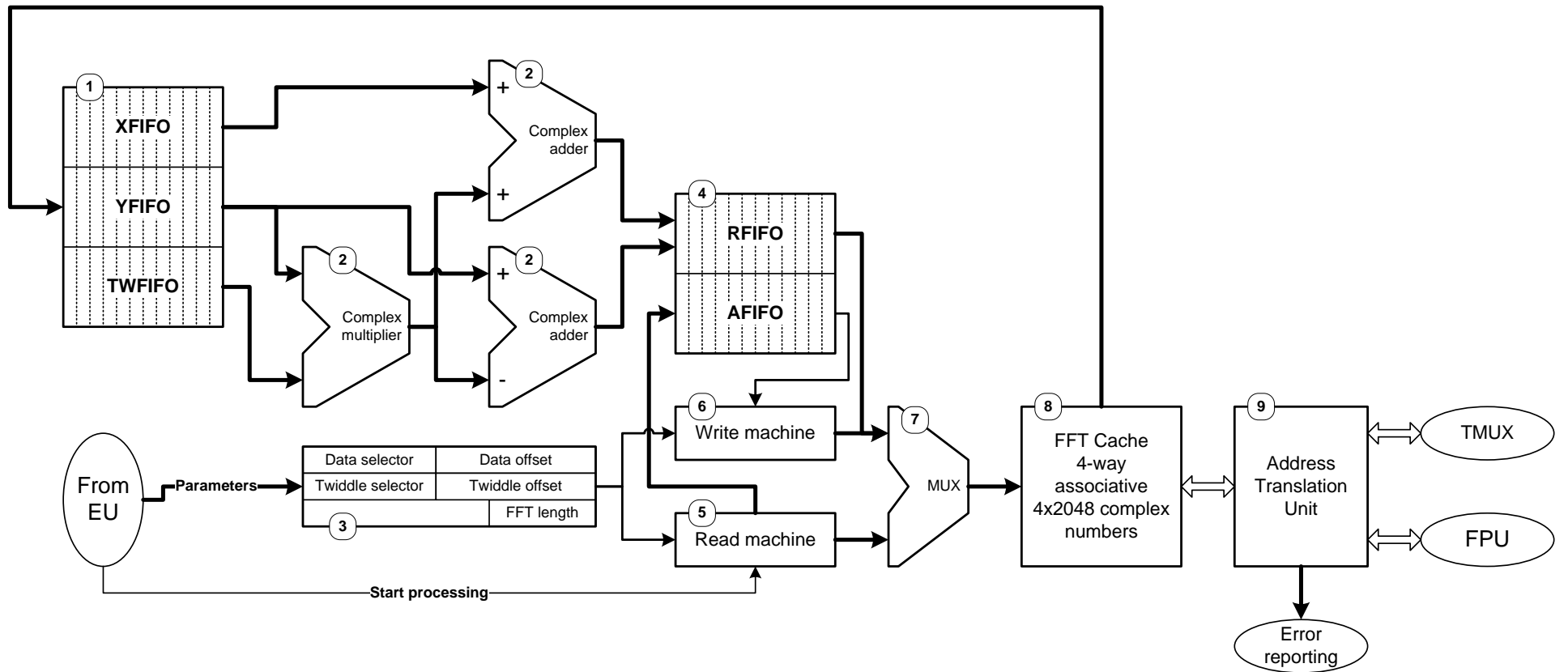
The control of completion of the execution of the FlyBy-instruction is carried out with the help of certain flags located in the predefined memory cells. For example, the completion of the FFT instruction is accompanied by the ASCII code of the string 'ENDOFFFT' written into the last 8 bytes of the data array.

At this time, the family of FlyBy instructions is represented by only one instruction - FFT.

FFT Machine

Fast Fourier Transform Machine performs processing of complex numbers, the real and imaginary parts of which are represented in a single-precision floating-point format.

The length of the data block is always a multiple of degree 2 and can range from $2 \wedge 1$ to $2 \wedge 20$. The block of twiddle factors always has a length of 2 times less than the length of the data.

XFIFO ①

YFIFO

TWFIFO

Complex adder ②

Complex multiplier ②

Complex adder ②

RFIFO ④

AFIFO

From EU

Parameters

Data selector	Data offset
Twiddle selector	Twiddle offset
③	FFT length

Write machine ⑥

Read machine ⑤

Start processing

MUX ⑦

FFT Cache 4-way associative 4x2048 complex numbers ⑧

Address Translation Unit ⑨

TMUX

FPU

Error reporting

FFT machine consists of the following units.

1. Input FIFO. Contains 3 64-bit buffers into which two source operands and one rotation factor are read from memory. The queues are designed so that all three source values are transferred to the FFT butterfly calculation at the same time.
2. Two adders and a multiplier for complex numbers calculate the FFT butterfly. The calculation is pipelined and allows you to get the result of a butterfly operation on each clock cycle. The length of the pipeline is equal to 7.
3. The registers containing the operation parameters are the logical address of the data block, the logical address of the twiddle factors block and the size of the data block.
4. Result queue. Two complex numbers of the result and address information defining where to write the results are placed in this queue.
5. The machine that controls the reading of the source data and twiddle factors.
6. The machine that controls the recording of results. Initiates write transactions if there is valid information in the result queue and the address queue. The contents of the address queue are used by the machine to determine the address for writing the contents of the results queue.
7. The transaction multiplexer transfers transactions from the write and read machines to the cache buffer. The priority is always to have a data write channel, which eliminates the situation of overflow of the entire pipeline and queues.
8. Cache buffer. Size 64 Kb or 8192 complex numbers. The cache is used to store twiddle factors and data. When writing data, the cache works in write-through mode, all transactions write intermediate data or final results are always translated to TMUX/FPU, regardless of whether the logical address was within the range of cached addresses or not. The cache uses a logical address to determine the presence of data in it.
9. Address translation unit. It loads the descriptors of two objects in which the processed data and twiddle factors are located. The unit controls the FFT machine's access to objects, checks the limits of the objects, reloads the segment descriptors if the object is segmented, and stops the FFT machine's operation with generating a violation message if an access error to the object is detected.
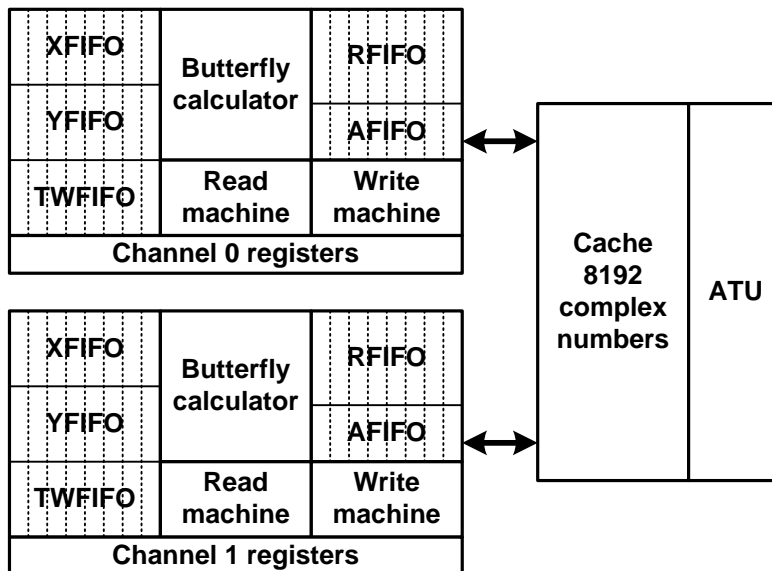
Data or twiddle factors can be located both in the local memory and in the memory of another core of the multiprocessor network. The address translation unit, depending on the location of the object, accesses either the local memory via TMUX or the memory of another core using FPU. Neither data, nor twiddle factors can be placed in objects described by stream descriptors.

The decimation-in-time based structure is used to compute FFT. The source data should already be written with a permutation of the even and odd sequences.
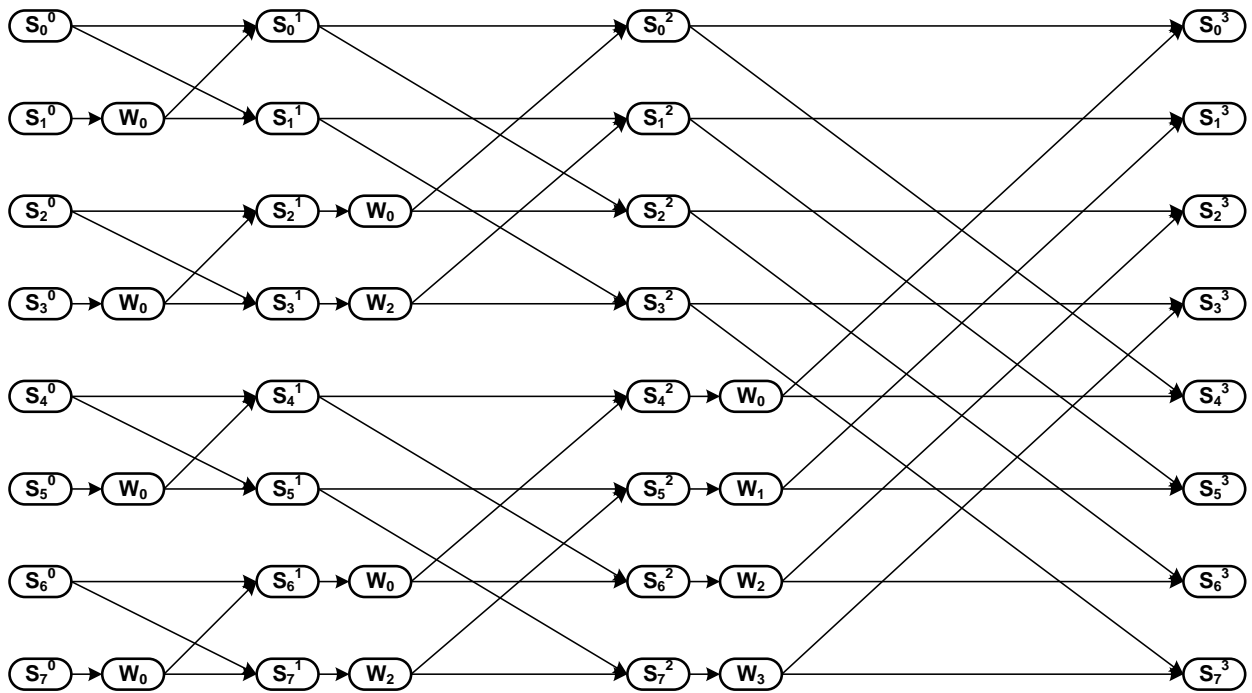
Dual FFT Core

A dual core FFT machine can simultaneously perform two FFT operations. The machine contains twice the number of input data queues and coefficients, two FFT butterfly calculators, two data read machines, two write machines and two results queues.

XFIFO	Butterfly calculator	RFIFO
YFIFO		AFIFO
TWFIFO	Read machine	Write machine
Channel 0 registers		

| XFIFO	Butterfly calculator	RFIFO
YFIFO		AFIFO
TWFIFO	Read machine	Write machine
Channel 1 registers		

Cache 8192 complex numbers | ATU

The data and twiddle factors cache, as well as the address translation unit, are common to the two channels of the FFT machine.

Processor X16x2

The processor contains two cores X16. The transaction multiplexer and stream controller of each core have 1 additional input to serve the transactions generated by the neighboring core.
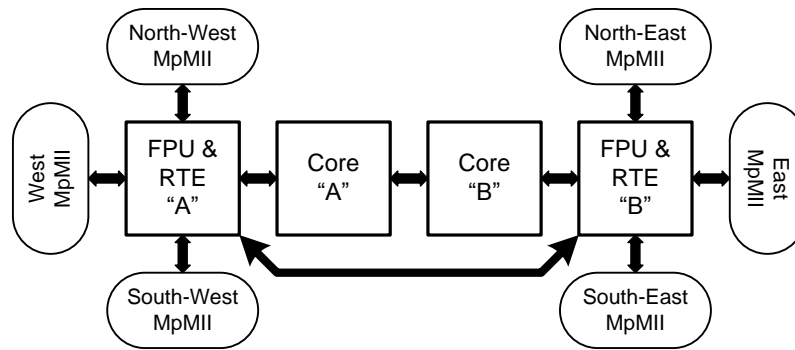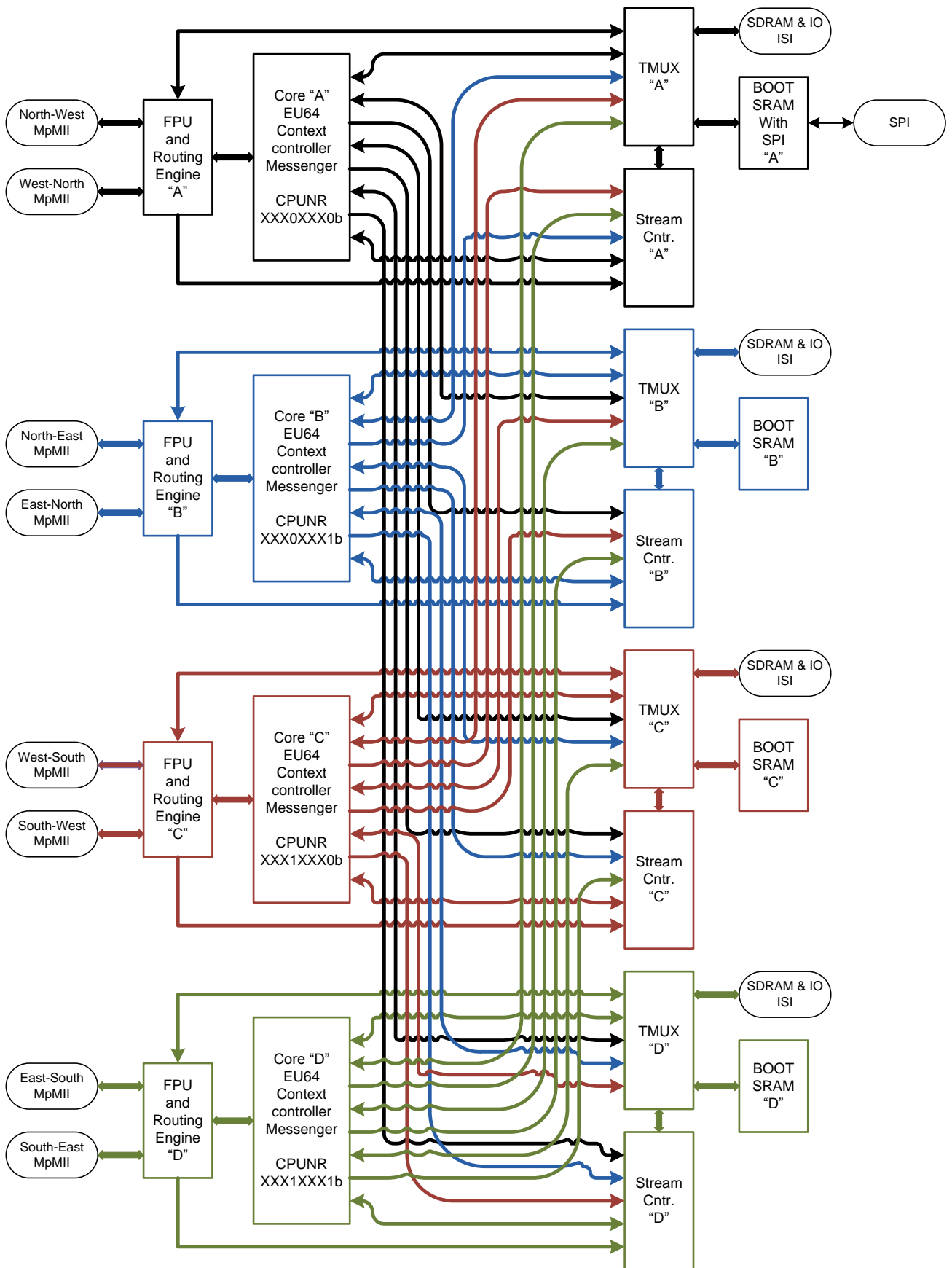
FPU is not used if data transactions are performed within a cluster, since all the cluster cores are connected to all transaction multiplexers and all stream controllers.

The network CPU numbers of the cores determine the control register belonging to the core "A", for core "B" the network number is formed in hardware. For example, if the core "A" network number is 20h, then the "B" core will have the number 21h.

Processor X16x4

The processor contains four X16 cores. The transaction multiplexer and stream controller of each core have 3 additional inputs to serve the transactions generated by the neighboring cores.

It should be noted that the data from the stream controller can only be read by the local core. All other cores, as well as FPUs, can only transfer data to streams. For example, streams hosted in the stream controller "B" are readable only for core "B". This is indicated by arrows on the connecting lines on the processor's diagram. A distinctive feature is also the fact that the SPI interface has only the core "A".

FPU is not used if data transactions are performed within a cluster, since all the cluster cores are connected to all transaction multiplexers and all stream controllers.

The network numbers of the cores determine the control register belonging to the core "A", for all other cores the network number is formed in hardware. For example, if the core "A" network number is 20h, then the "B" core will have the number 21h, the "C" core is 30h and the "D" core is 31h.

The 4-core cluster has 8 MpMII interfaces. The following block diagram explains the configuration of the cluster.
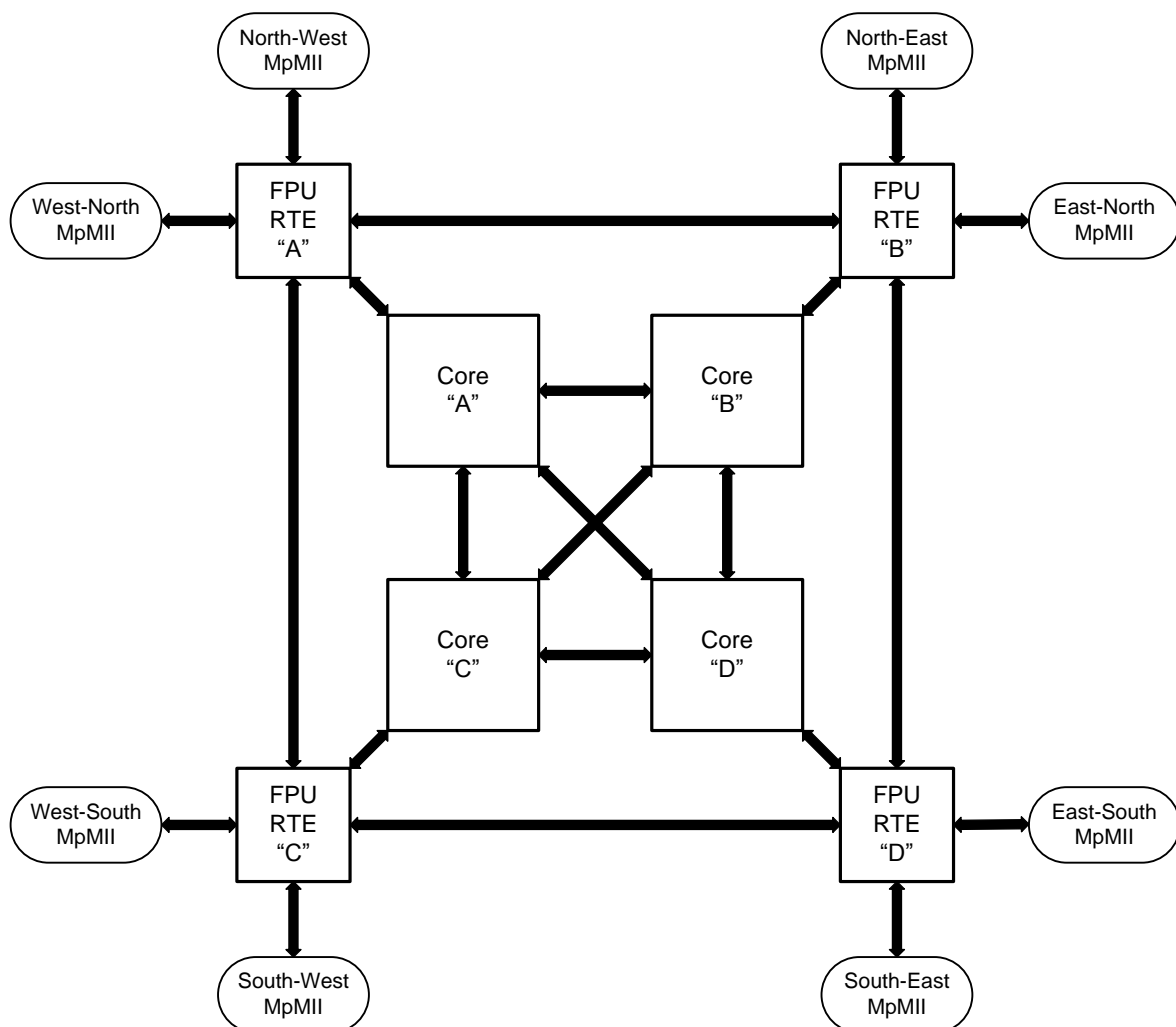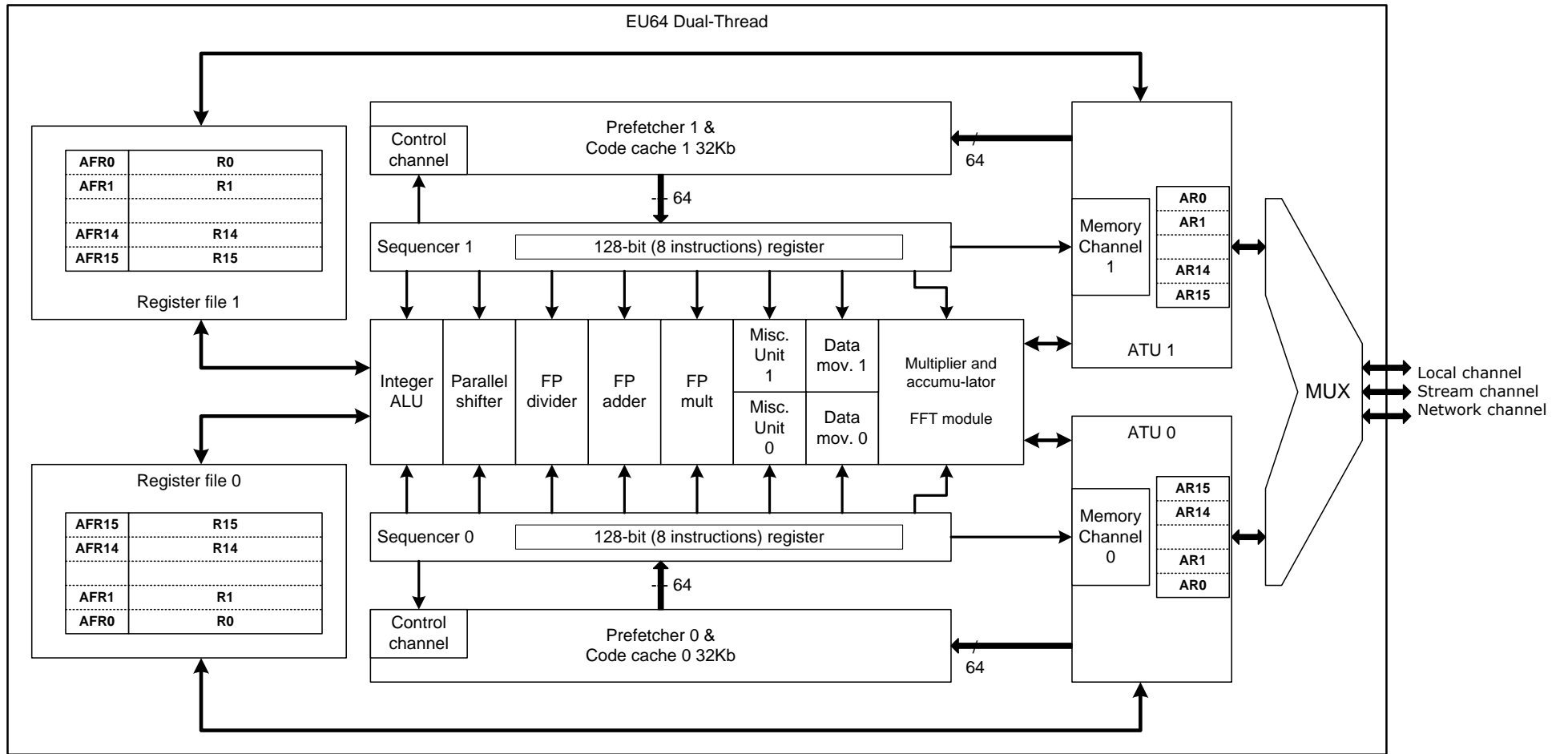

X16DT Dual-Thread core

A two-thread core is capable of simultaneously executing two instruction streams. At the same time, there is no need for special software support for the

functioning of such a core, unlike multi-core systems, where each core is treated as a separate processor and the operating system requires some software support for multiprocessing. Threads can be interrupt handlers, message processing routines, and can be parallel executable processes. The two-thread core allows more full use of the computing resources of the processor. For example, while one thread processes floating-point numbers using multiplication/addition/division blocks, the second thread can use resources to perform integer arithmetic or logical or shift operations without any restriction. The differences between single-threaded and double-threaded cores are as follows.

TMUX has a dual set of system registers CSR, CPSR and PTR. Each thread has access only to its own set of registers. The context controller and the message controller can switch between sets of registers.

Context and message controllers have hardware support for two channels for interacting with two threads, as well as modified firmware.

The executive unit has two instruction prefetchers, two caches of program code, two sequencers, two sets of registers R, AR and AFR, two blocks of execution of miscellaneous instructions, two address translation units.

EU64 Dual-Thread

Prefetcher 1 &
Code cache 1 32Kb

Control
channel

/ 64

-- 64

Sequencer 1 | 128-bit (8 instructions) register

AFR0	R0
AFR1	R1
AFR14	R14
AFR15	R15

Register file 1

Memory
Channel
1

| AR0 |
| AR1 |
| AR14 |
| AR15 |

ATU 1

Integer
ALU

Parallel
shifter

FP
divider

FP
adder

FP
mult

Misc.
Unit
1

Data
mov. 1

Multiplier and
accumu-lator

FFT module

Misc.
Unit
0

Data
mov. 0

MUX

Local channel
Stream channel
Network channel

ATU 0

Memory
Channel
0

| AR15 |
| AR14 |
| AR1 |
| AR0 |

Register file 0

AFR15	R15
AFR14	R14
AFR1	R1
AFR0	R0

Sequencer 0 | 128-bit (8 instructions) register

-- 64

Control
channel

Prefetcher 0 &
Code cache 0 32Kb

/ 64

Common blocks are multiplier-battery, FFT machine, FP multiplier, FP divider, FP adder, parallel shifter, integer ALU. These blocks are used in time-sharing mode with rotating priority between threads. If both streams continuously use an integer ALU on each clock cycle, they will receive access alternately.

After the processor is initialized by the system reset signal, only one thread starts to execute the code (thread 0). Thread 1 is activated if there are selectors of two or more processes in the table of executable processes.

T2X16T2X32 Quad-Thread core

T2X16T2X32 core contains 2 execution units in X16 configuration and two units in X32 configuration. The leader (starting after a system reset) is one of the X16 blocks, respectively, and the Kernel is used in the version of the X16 instruction set. The processor can execute simultaneously 2 processes in the X16 instruction set and two processes in the X32 instruction set. The processor can execute mixed instruction set processes that contain code objects in the X16 instruction set and code objects in the X32 instruction set.

In-System Interface

The interface is intended for connections of the type "master" - "slave". It is synchronized with the clock signal. It can transmit information at each clock cycle from the master to the slave and vice versa. Commands, addresses and data are sent from the master to the slave. The data is valid if the data is written to the internal resource of the slave device. The slave can only return the read data to the master.
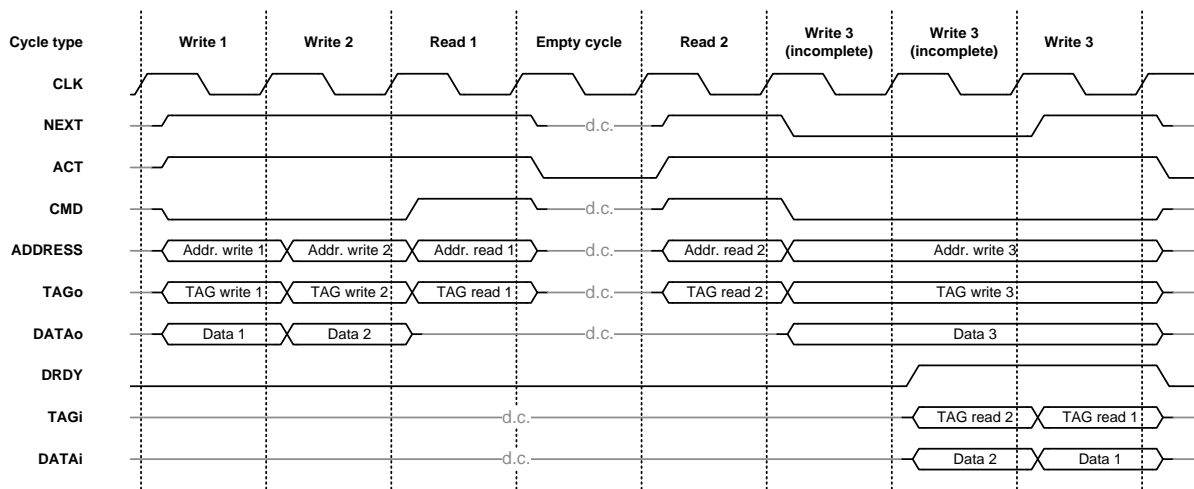
In general, the ISI consists of the following lines.
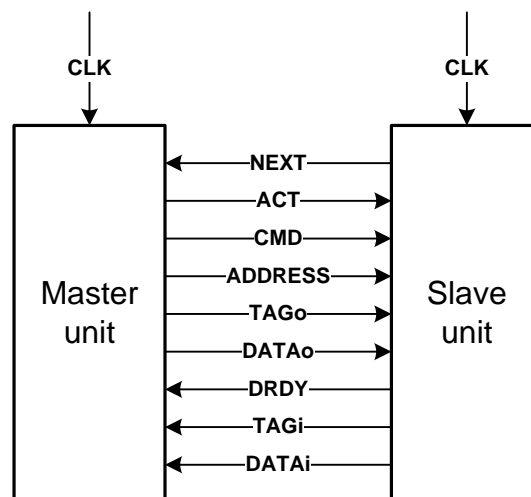
1. ACT - transaction activation signal. The master sets ACT = 1 when the transaction is active.
2. SMD - the type of transaction - read or write. CMD = 0 specifies the write transaction.
3. ADDRESS - the address describing where in the slave device it is necessary to place the data or where to read it.

4. TAGo, TAGi - transaction tag. Used only in read transactions. This tag must be returned by the slave device at the same time as the data read. By the transaction tag, the master determines the location in which the read data is to be placed, for example, the tag can contain a general-purpose register number.
5. DATAo - data that the master sends to the slave. The value is valid only for CMD = 0.
6. NEXT is a signal that notifies the master that a transaction can be received for processing in the current clock cycle. NEXT = 1 allows the master to set the parameters of the new transaction in the next clock cycle. NEXT = 0 indicates that the current transaction must be repeated in the next clock cycle.
7. DRDY - data availability. The slave notifies the master that the data requested previously is present on the DATAi bus.
8. TAGi – tag of the read transaction.
9. DATAi - data transferred to the master.

An example of the time diagram of the ISI.

Cycle type	Write 1	Write 2	Read 1	Empty cycle	Read 2	Write 3 (incomplete)	Write 3 (incomplete)	Write 3
CLK								
NEXT				d.c.				
ACT								
CMD				d.c.				
ADDRESS	Addr. write 1	Addr. write 2	Addr. read 1	d.c.	Addr. read 2		Addr. write 3	
TAGo	TAG write 1	TAG write 2	TAG read 1	d.c.	TAG read 2		TAG write 3	
DATAo	Data 1	Data 2		d.c.			Data 3	
DRDY								
TAGi				d.c.		TAG read 2	TAG read 1	
DATAi				d.c.		Data 2	Data 1	

The sequence of data returned by the slave module may not coincide with the order of the read requests generated by the master module. The only way to determine which response to which read transaction has been sent to the master module is the transaction tag. In CoreOne and CoreQuad cores ISI is used everywhere. It connects EU and TMUX, EU and FPU, FPU and stream controller, FPU and TMUX, etc.

Two ISIs from each core are output as external interfaces, intended for connection of the memory controllers and modules of peripheral input / output equipment.

Bus	Description
Common buses	
CMD	Operation type. 1 – read, 0 – write.
ADDR[44:0]	Address of the location of the least significant byte of the data element to be transmitted.
DATA[63:0]	The data transferred to the SDRAM or IO device.
BE[7:0]	The lines that determine which bytes on the DATA bus are valid.

	Encoded according to the table:			
	ADDR[2:0]	Location on the DATA bus	BE[7:0]	
	8-bit data			
	000b	[7:0]	11111110b	
	001b	[15:8]	11111101b	
	010b	[23:16]	11111011b	
	011b	[31:24]	11110111b	
	100b	[39:32]	11101111b	
	101b	[47:40]	11011111b	
	110b	[55:48]	10111111b	
	111b	[63:56]	01111111b	
	16-bit data			
	00Xb	[15:0]	11111100b	
	01Xb	[31:16]	11110011b	
	10Xb	[47:32]	11001111b	
	11Xb	[63:48]	00111111b	
	32-bit data			
	0XXb	[31:0]	11110000b	
	1XXb	[63:32]	00001111b	
	64-bit data			
	XXXb	[63:0]	00000000b	
TAG[20:0]	Tag of transaction.			

SDRAM ISI	
SDNEXT	The readiness of the SDRAM interface unit to accept the transaction.
SDACT	Activation of transaction to SDRAM.
SDCMD	Command, read/write.
SDADDR[41:0]	Address of the 64-bit word.
SDBE[7:0]	Byte enables.
SDDI[63:0]	Data to the SDRAM.
SDTI[1:0]	Tag.
SDDRDY	Readiness of data from SDRAM.
SDDO[63:0]	Data read from the SDRAM.
SDTO[1:0]	Tag read from SDRAM.
IO ISI	
IONEXT	IO resource readiness to accept the transaction.
IOACT	Activation of transaction to IO resource.
IODRDY	Readiness of data from IO resource.
IODAT[63:0]	Data read from the IO resource.
IOTAG[20:0]	Tag read from IO resource data.

Transaction tag encoding.

```
 20    18 │1716│ 15    13│ 12                                    0
```

TAG from module EU/FPU/Messenger/Strem controller/Context controller

Channel:
000 – EU
001 – FPU
010 – Stream controller
011 – Context controller
100 – Messenger
Only for CoreQuad processor
101 – Neighborhood core
110 – Neighborhood core
111 – Neighborhood core

Operand size:
00 – 8-bit
01 – 16-bit
10 – 32-bit
11 – 64-bit

Copy of ADDR[2:0]
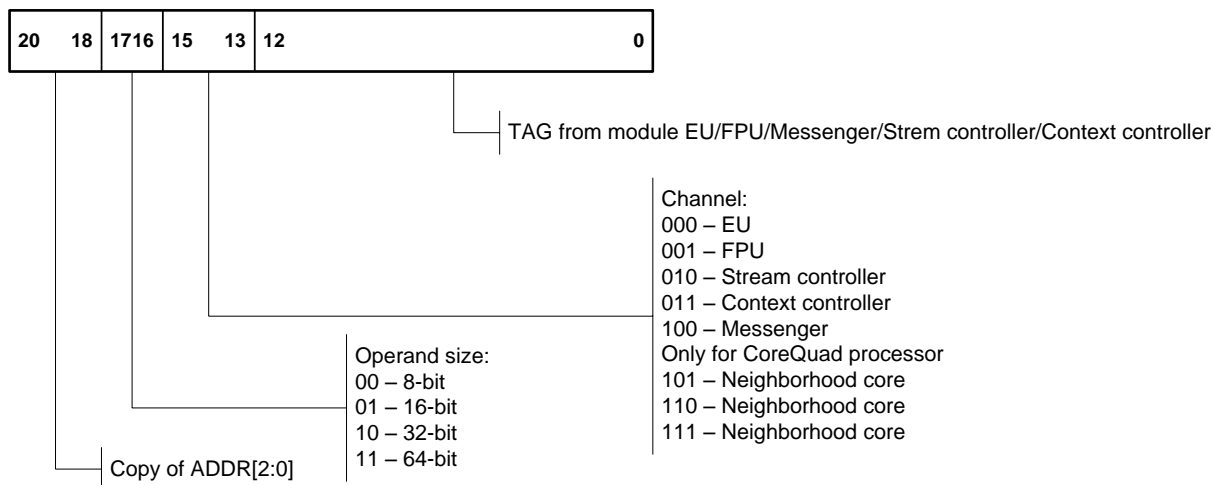
The bit field of the operand size and the copy of the lower three bits of the address are used to determine the SDDAT or IODAT bus lines containing the read data.
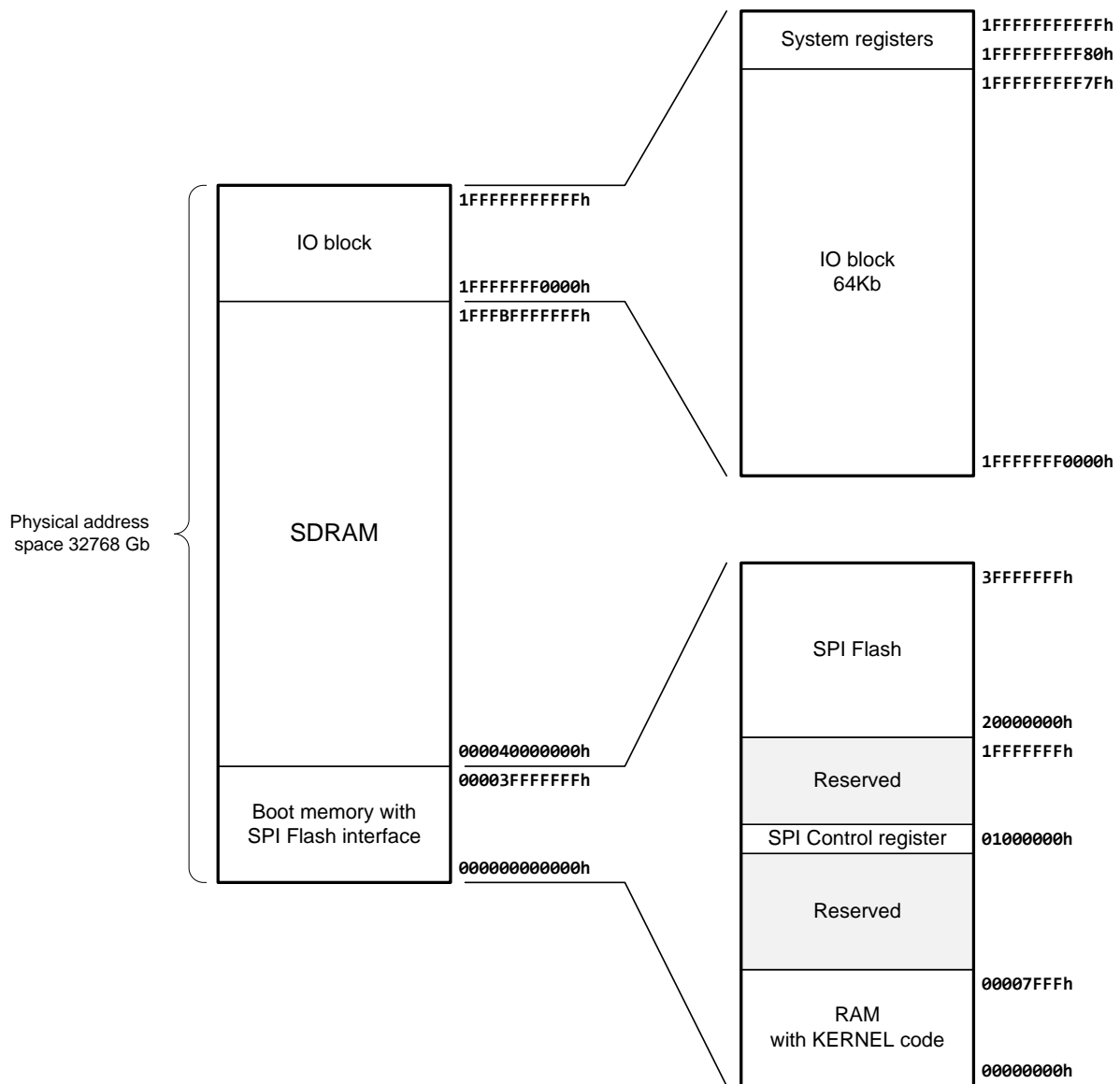
MpMII

The interface is intended for communication between the RTE module and the physical implementation module of multiprocessor communication. It is synchronized by the main clock signal.

Bus	Description
NETSTBO	Strobe of data output. 1 - there are valid data on the NETDO bus.
NETDO[32:0]	A bus for transferring data to an adjacent processor. Data is transmitted in bits [31: 0] Bit 32 is used to refer to the first 32-bit data word in the sequence. For the first word, the bit is set to 1, the remaining words are followed by 0.
NETSTBI	Strobe of data reception. 1 - there are valid data on the NETDI bus.
NETDI[32:0]	A data bus received from a neighboring processor.

Address space configuration

An address space of 1 GB, starting with a zero address, is allocated to accommodate devices that contain the boot code and system kernel software. Directly from the zero address, the built-in memory containing the initial initialization code is allocated. The total size of this block of memory is 32KB. The top of the buffer size of 1024 bytes is designed to organize packet data exchange and control information with an external SPI device. But the same block can be used as a RAM.

IO block

1FFFFFFFFFFFh

1FFFFFFFF0000h

1FFFBFFFFFFFh

SDRAM

Physical address
space 32768 Gb

000040000000h

00003FFFFFFFh

Boot memory with
SPI Flash interface

000000000000h

System registers — 1FFFFFFFFFFFh / 1FFFFFFFFF80h / 1FFFFFFFFF7Fh

IO block
64Kb

1FFFFFFF0000h

SPI Flash — 3FFFFFFFh

20000000h

Reserved — 1FFFFFFFh

SPI Control register — 01000000h

Reserved

RAM
with KERNEL code — 00007FFFh

00000000h

At address 1000000h, the SPI interface control register is located. Writing to it lowest 16-bit word initiates the procedure of transfer a variable number of bytes located from address 7C00h to the SPI FLASH. The transferred data length is programmed by the contents of the control register.

31 24	23 16	15 10	9 0
Not used	RxCMD	Not used	TxCNTR
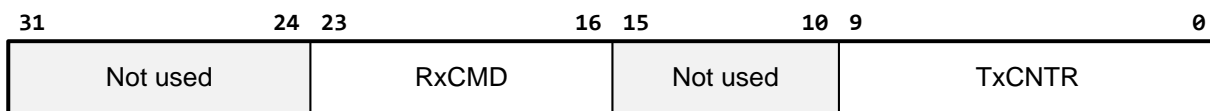
The counter of the bytes transferred to the SPI is located in bits [9:0]. In the bits [23:16] there is a read command, which must be transferred to the SPI Flash during read transactions. Read transactions is performed, when processor reads data from SPI Flash address space. SPI Flash address space starts from 20000000h and ends at the 3FFFFFFFh. Currently SPI Flash controller supports instructions with a several codes.

RxCMD	Instruction
05	Read status register 1
35	Read status register 2
15	Read status register 3
03	Read data
AB	Release ID
90	Read manufacturer/device ID
0B	Fast read
4B	Read unique ID
5A	Read SFDP Register
48	Read security register

For IO devices, a 64KB memory block is allocated. The last 128 bytes of the block are used to place the system status and control registers.