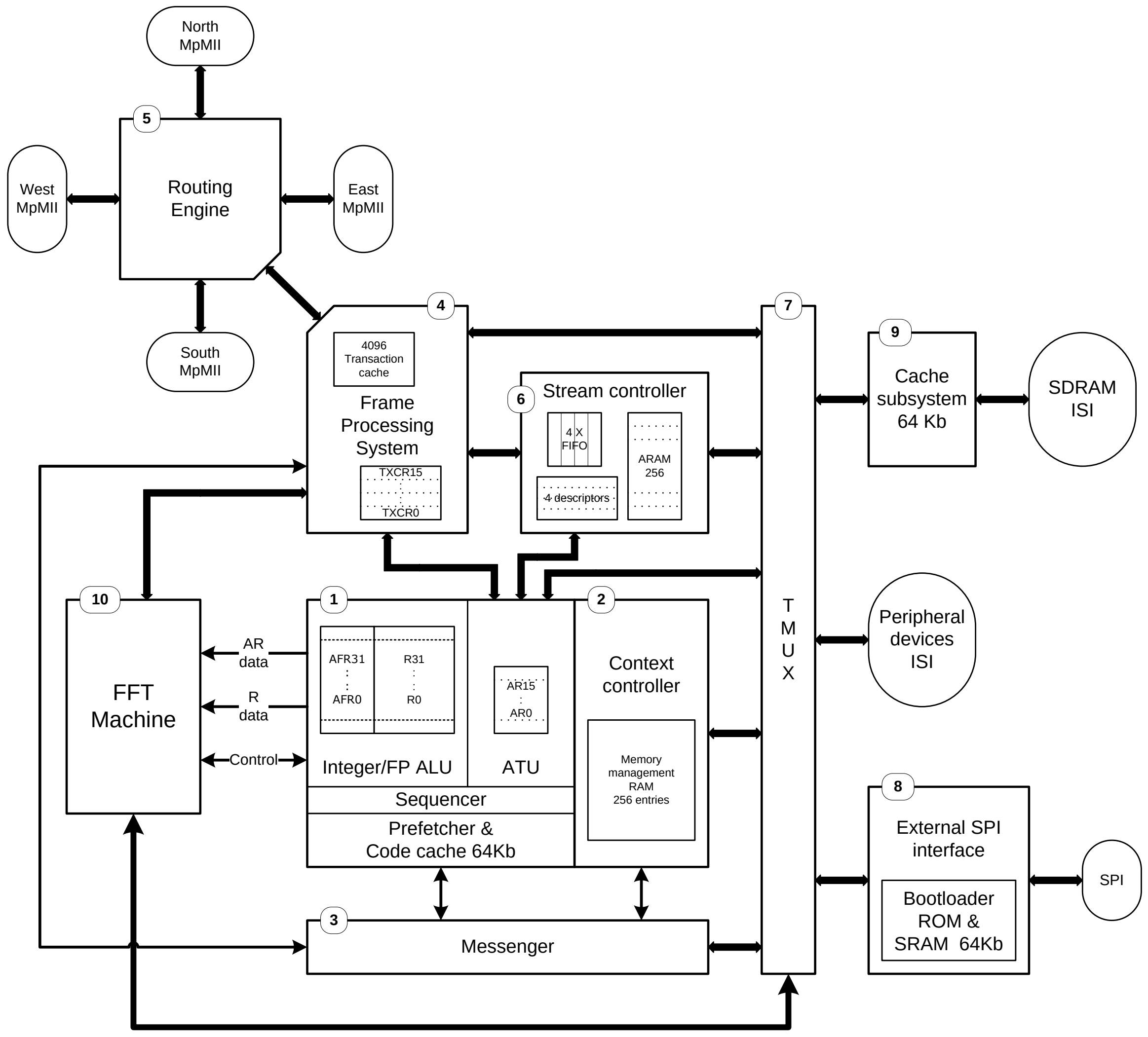
Contents

Processor architecture.

CoreOne32V0.

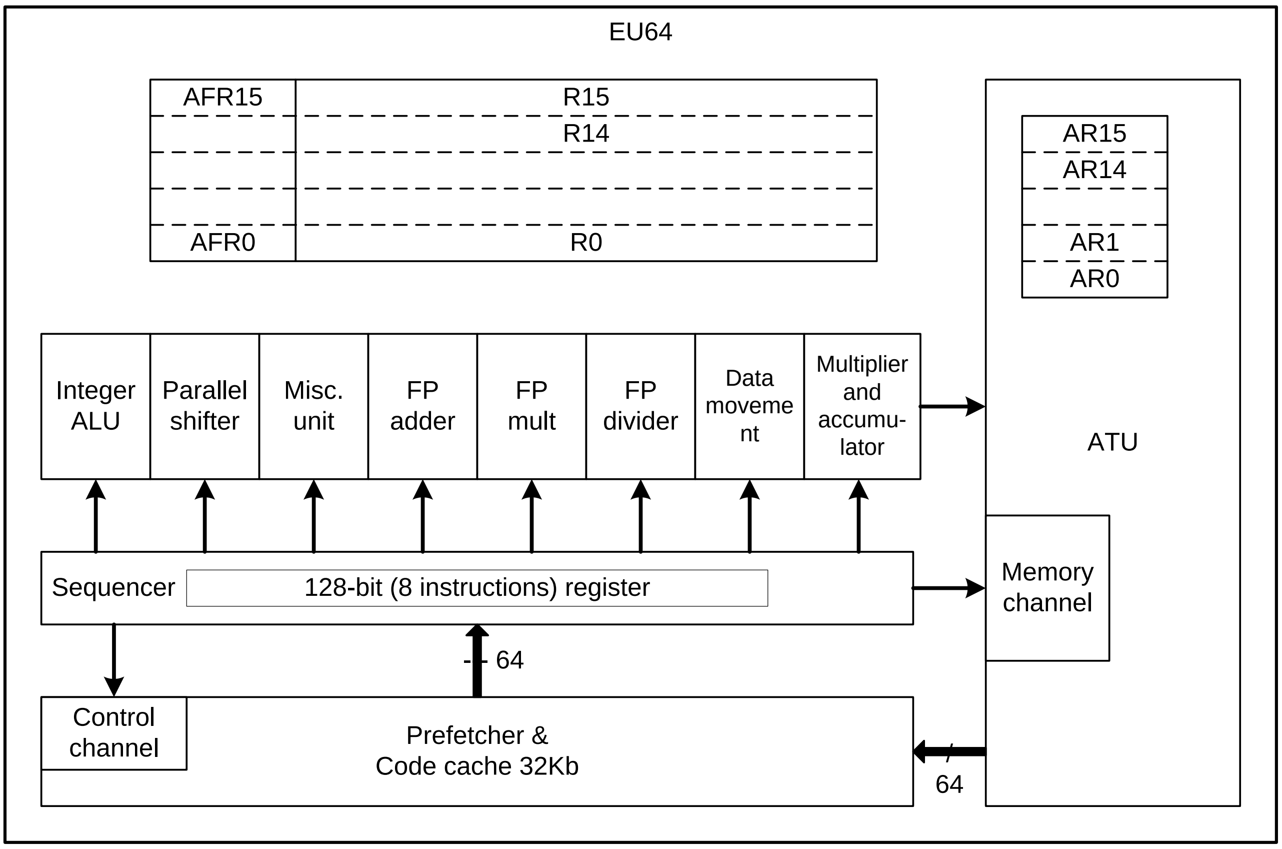


1. The executive unit, capable of running up to 8 instructions at a time. The EU consists of:
   - Arithmetic logic unit containing 32 128-bit general-purpose registers - R[31:0] and identified with them 32 32-bit flags registers AFR[31:0];
   - Address translation unit containing 16 address registers 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.

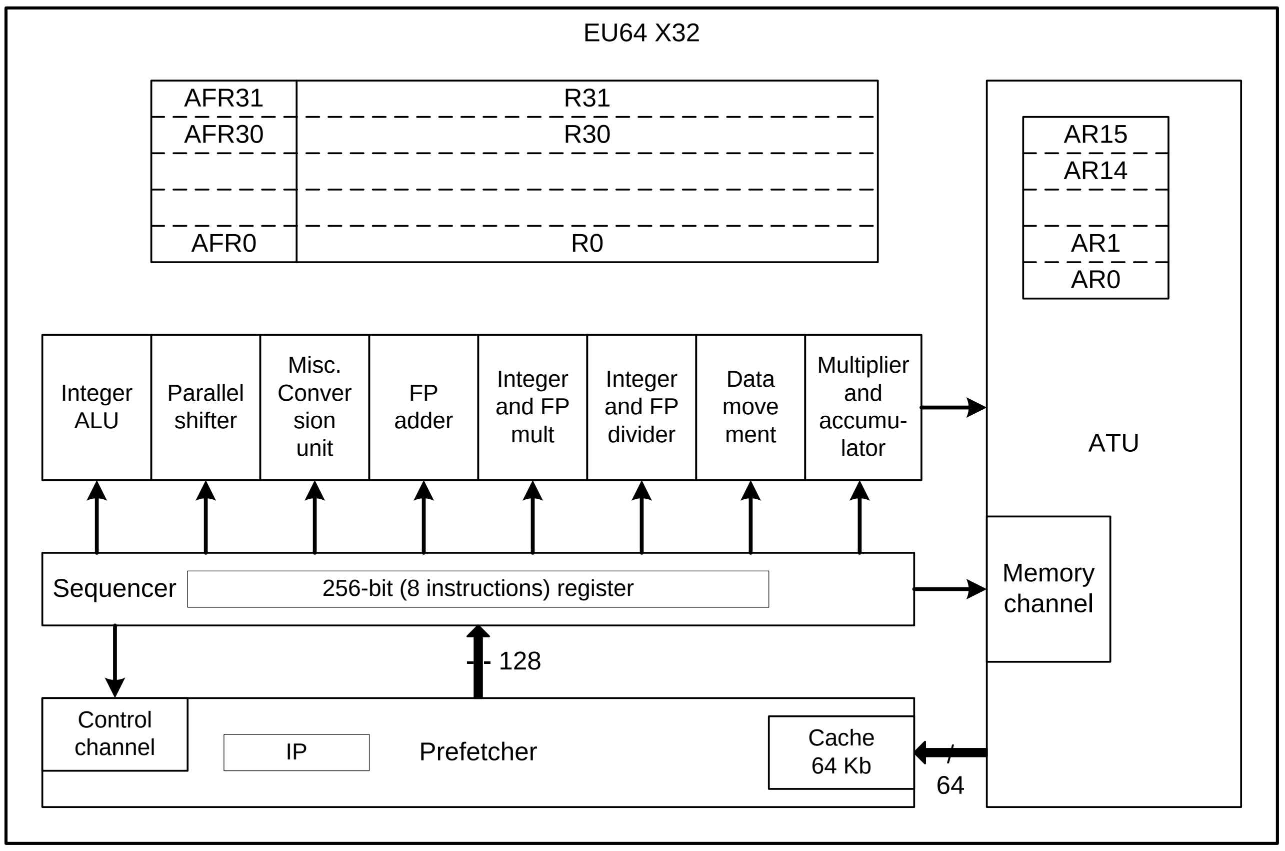
3. The messenger is the system responsible for setting messages in the process message queue and for immediately calling procedures for handling exceptions, hardware interrupts, and debugging interrupts.
4. The Frame Processing Unit (FPU) is the unit responsible for forming frames for the multiprocessor 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 CoreOne32V0 and CoreOneV0.

The main differences between the 32V0 and V0 cores are in the structure of the execution units of the processor. Also core V0 has 32Kb of boot SRAM and core 32V0 has 64Kb. EU V0 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.



EU 32V0 contains a 10-channel sequencer also. Unlike from V0 core a 32V0 core contains divider and multiplier which can process integer operands. Divider and multiplier in core V0 can process only floating point data. Miscellaneous operations unit in the 32V0 core is able to convert integer to floating point and vice versa. Core 32v0 prefetcher feeds sequencer by 128-bit bus with throughput up to four instructions per clock. Also prefetcher 32V0 has 64Kb space of the internal instruction cache.



A lot of fundamental differences between CoreOneV0 and CoreOne32V0 are in the instruction set. CoreOne32V0 has:

1. Data conversion instructions "integer <-> Floating point".
2. Integer division and multiplication instructions. They able to process signed and unsigned integer numbers.
3. Direct reference of address mode, operand size and additional short displacement in the instruction opcode.
4. Direct reference of the index of the 16-bit word which loaded into address or data register from an appropriate instruction field.
5. CALL and JUMP instructions with immediate displacement to destination instruction.
6. Extended set of instructions what used for address registers modification. These instructions are designed to allow to loading address registers by immediate data or modify it by sum with immediate data.

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.

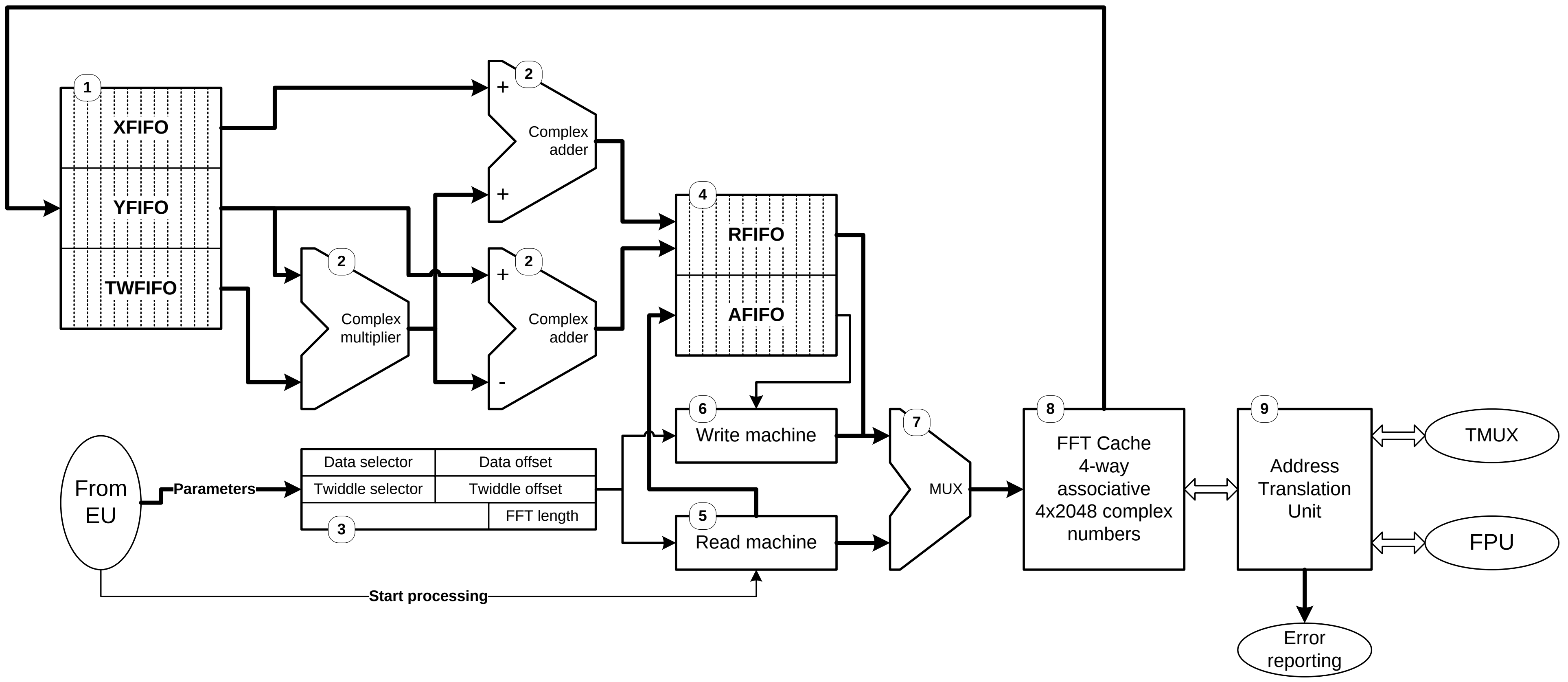
The control of completion of the execution of the FlyBy-instruction is carried out with the help of certain flags located in the predefined memory cells. For example, the completion of the FFT instruction is accompanied by the ASCII code of the string 'ENDOFFFT' written into the last 8 bytes of the data 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 ^ 1 to 2 ^ 20. The block of twiddle factors always has a length of 2 times less than the length of the data.

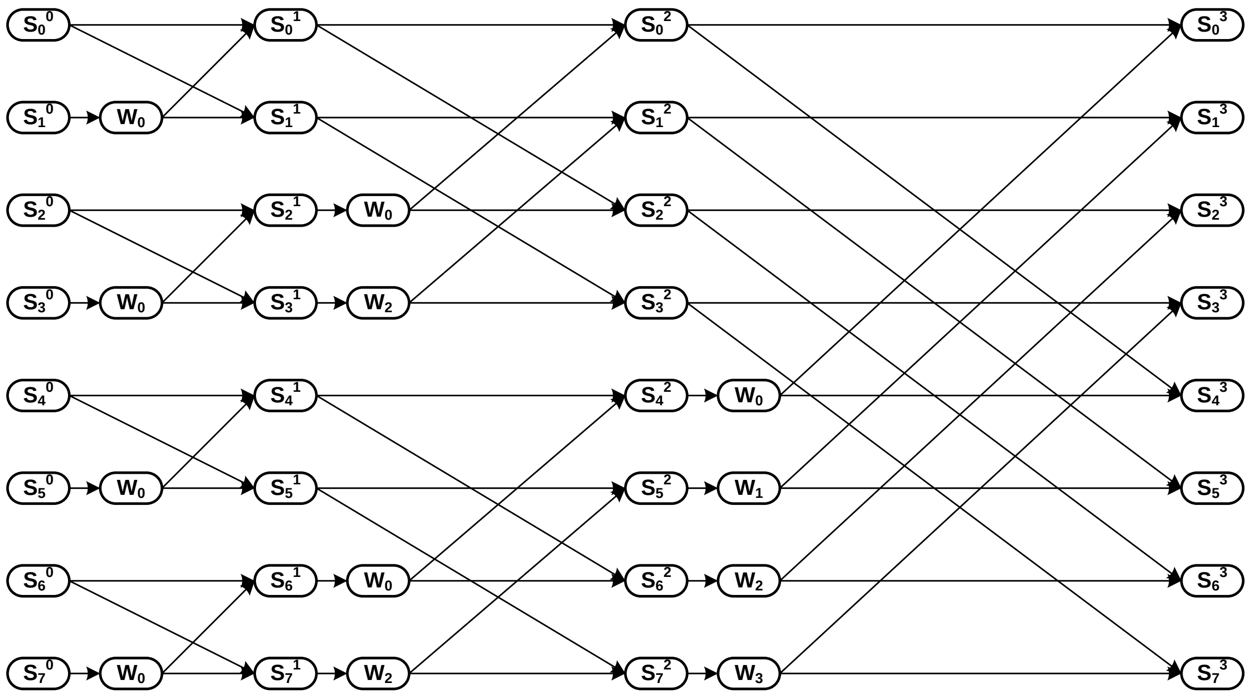

1
XFIFO
YFIFO
TWFIFO
2
+
+
Complex adder
2
Complex multiplier
2
+
-
Complex adder
4
RFIFO
AFIFO
6
Write machine
5
Read machine
7
MUX
8
FFT Cache 4-way associative 4x2048 complex numbers
9
Address Translation Unit
TMUX
FPU
Error reporting
From EU
Parameters
Data selector
Data offset
Twiddle selector
Twiddle offset
FFT length
3
Start processing

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.

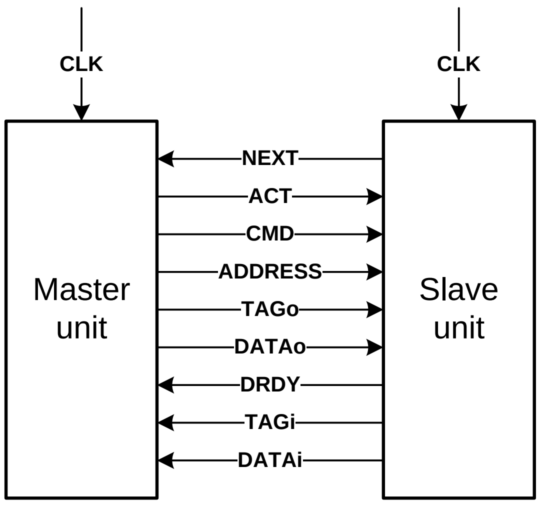
The decimation-in-time based structure is used to compute FFT. The source data should already be written with a permutation of the even and odd sequences.



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.

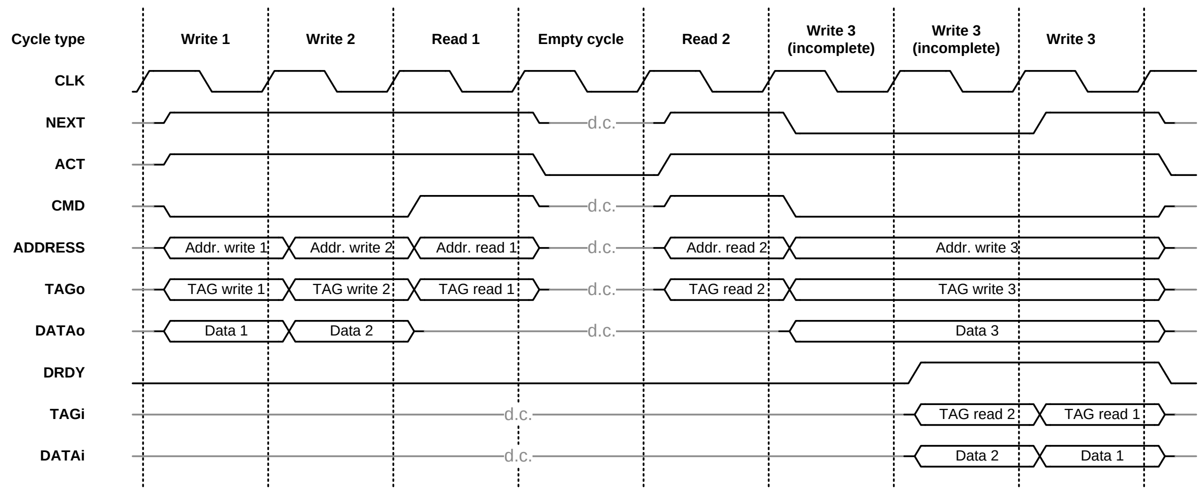
In general, the ISI consists of the following lines.



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.



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.

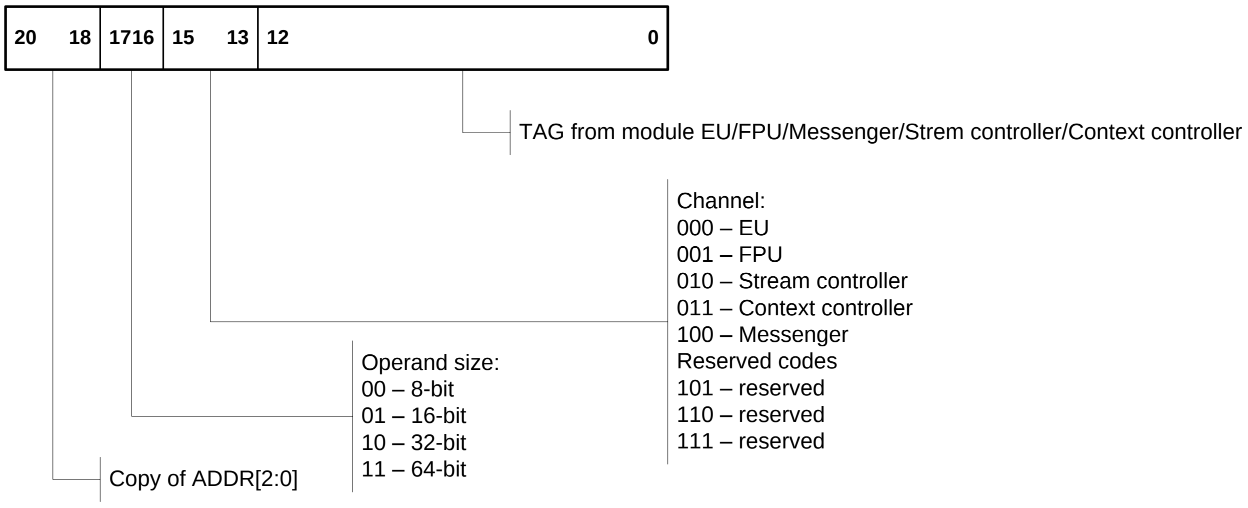
Bus	***Description***
Common buses	
CMD	Operation type. 1 – read, 0 – write.
ADDR[44:0]	Address of the location of the least significant byte of the data element to be transmitted.

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.



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.

EU tag format:

8	7	6	5	4	3	2	1	0	
0	LDO Inst	PUSH/ POP 3-d cycle	Sec. Cycl e	REG					LD/ST/PUSH/POP operations
1	0	0	0	Selector register index			QWord index		Descriptor load operations
1	0	1	0	LEF	CALL RET	Prefetcher tag			Prefetcher operations
1	1	0	0	0	0	0	0	TAG	FPMULACC tag
1	1	1	TAG						FlyBy operations

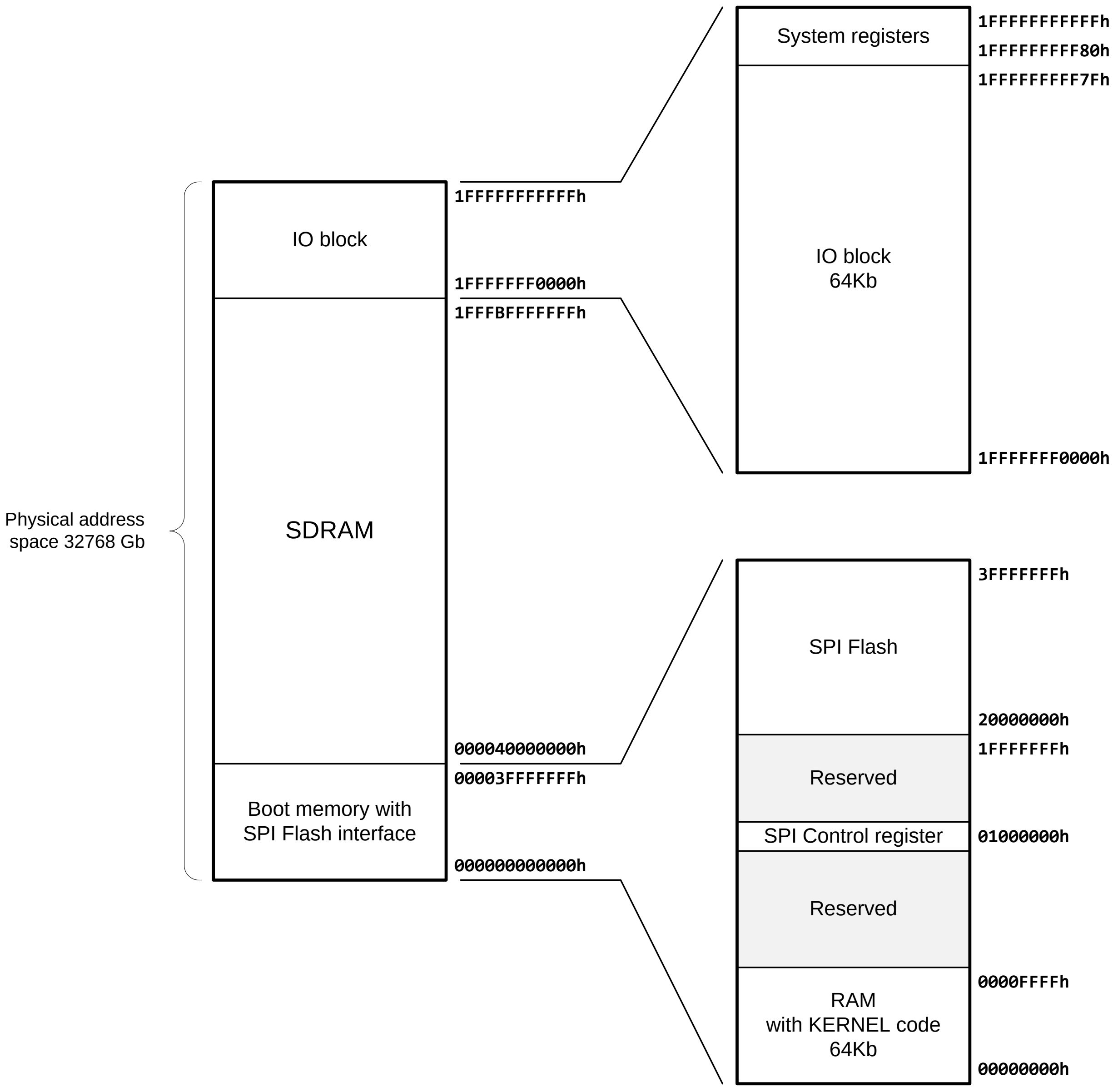
MpMII

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.



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 0FC00h 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.