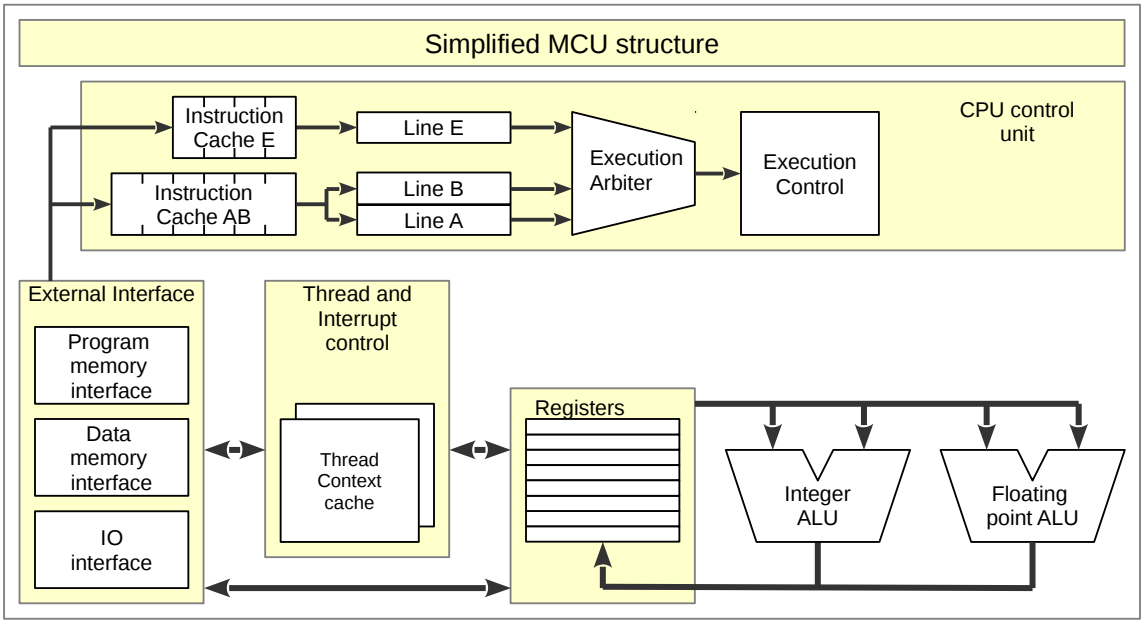
32-bit microcontroller core
Product overview

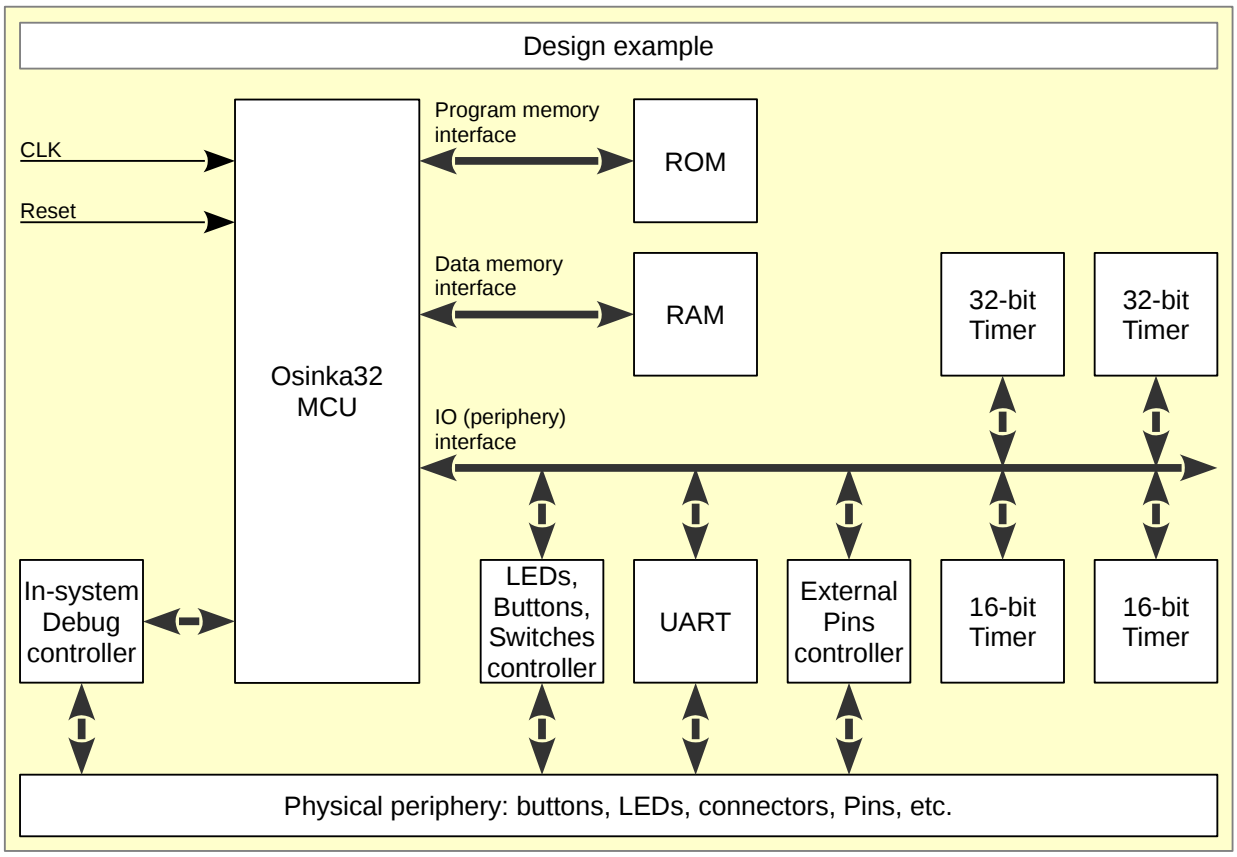
Osinka32 is a 32-bit microcontroller core with a native multithreading support. It is able to execute up to 2 commands per clock. It is able to detect execution branches in a very early stage and execute them in zero clocks. Osinka32 operates with 8, 16 and 32-bit integer data and 32-bit floating point data.



Main features

- Integer 8, 16 and 32-bit operations, 32-bit floating point operations
- Executes up to 2 commands per clock
- Hardware multi-threading support
- Branch (JMP) instructions in general require 0 clock cycles for execution, no penalty
- Switch to IRQ in general requires 0 clock cycles
- Separate code, data and IO spaces
- Free IDE tools with in-system debugger

Osinka32 has separate address space for executable program, data and IO. This allows better performance and parallelism.



Approximate size and CLK max for different FPGAs		
FPGA	% of usage	Cells
Altera Cyclone IV EP4CE22F17C6	40	~9000

Performance
In order to cope with some peripheral latencies osinka32 can postpone some operations and continue execution until necessary data becomes available. This is very useful when reading data from periphery: whenever possible osinka32 will not stop execution of program if data reported not to be ready.

Floating point operations require more than 1 clock cycle for execution. The execution of program will not stop while FPU is running.

Osinka32 has a high performance algorithm of early branch detection. In general JMP instructions will not require any extra clock cycles if executed or if skipped.

Multithreading support
Multithreading is like having a C-program with several main() functions. Application can switch between them using a high-performance mechanism supported by hardware. This switch is really fast and takes ~3..5 clock cycles to switch from one thread to another, including penalty for refilling an instruction cache. Hardware takes care of thread context saving and retrieving which is performed automatically in the background.

Threading is very useful when application has to wait for some external events. It gets much easy to make application non-blocking. In case of waiting one thread can release CPU and let other threads run.

Interrupts
Unlike many other microcontrollers, osinka32 uses a thread context (and not a stack) to save registers when an interrupt service routine starts. This reduces interrupt reaction time to 4 clock cycles. These 4 cycles are not lost: osinka32 will run main-process pipe-line and ISR pipe-line in parallel. A switch from main process to ISR will be done instantaneously: on the 5th clock cycle we will see a result of a first command of ISR.

ISR software does not need to save registers, they will be saved in a thread context by the hardware. Registers will be restored when ISR ends. This simplifies a software routine and increases the performance.

FPU
Osinka32 supports 32-bit floating point operations. Any of 32-bit registers can operate with integer and floating point data. Floating point ALU and integer ALU work in parallel.

IDE
Free IDE includes assembler, linker and in-system debugger. IDE is available for Windows and Linux (optionally for Mac). In-system debug controller uses UART interface and is capable to detect baud-rate automatically. It allows to program memory, set breakpoints, start/stop execution, see registers, etc.