The AJIT processor family

Madhav Desai
Department of Electrical Engineering
IIT Bombay

January 19, 2024

Background

- Started working on a processor in the 2016-2017 time-frame
  - Funding from MeiTY and Powai Labs.
- Leverage work done at EE (IITB) on AHIR-V2 tools.
  - Algorithm to hardware tool-set.
  - Successfully used in AJIT, NAVIC, AI/ML accelerator, Network router projects at IITB.

A wide range of processors has been developed and validated

- Single core, single thread: base general purpose processor for embedded.
  - 64-bit ISA, with vector instructions.
- Single core, dual thread: double the performance at marginal extra cost.
  - Validated on FPGA.
  - 1.9X the performance (matrix multiplication) with 1.25X area overhead.
- Multi-core, single/dual thread per core: high performance embedded applications.
  - Quad-core, eight thread (4x2x64) configuration validated on FPGA.

The ISA: SPARC V8

- Originally developed at SUN micro-systems.
  - Open-sourced by SUN, almost became IEEE standard (draft IEEE standard 1754).
- General purpose, RISC load/store, with floating point support, co-processor support, standard reference memory management unit.
  - RISC ISA, similar to ARM etc., but with two quirks, register windows and delayed control transfer.
- Proven use in mainstream computing, good software support.
- Currently SPARC-V8 ISA based LEON processors are used by the European Space Agency for fault-tolerant and reliable computing.

ISA

- Windowed Register File, 32 visible registers.
  - Quirk 1: register windows.
- Arithmetic/Logic instructions.
- Floating point instructions.
- Condition codes (integer and floating point).
- Control transfer instructions: branch, call, jump, return from trap.
  - Quirk 2: delayed control transfer.
- Miscellaneous instructions: save, restore, software-trap instructions etc.
- 16-level interrupt and vectored trap handling.
- Supervisor/user modes, with memory protection mechanisms.

Visible Registers

```
g0,g1, ... g7   globals

o0,o1, ... o7   outs
l0,l1, ... l7   local      register window
i0,i1, ... i7   ins
```

Quirk 1: Register windows

```
Window 0  outs  ==  Window 7 ins
Window 1  outs  ==  Window 0 ins
Window 2  outs  ==  Window 1 ins
Window 3  outs  ==  Window 2 ins
Window 4  outs  ==  Window 3 ins
Window 5  outs  ==  Window 4 ins
Window 6  outs  ==  Window 5 ins
Window 7  outs  ==  Window 6 ins

Notes: locals of each window are distinct.
```

Quirk 1: Register windows

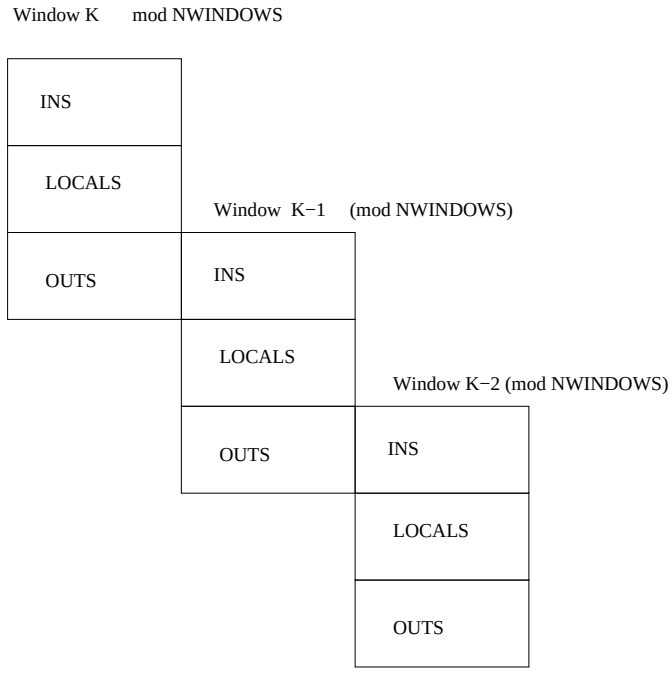


Figure: Overlapping register windows

Quirk 2: delayed control transfer

SPARC-V8

```
-----------------------------------------
PC := 0x0 NPC := 0x4
while (1) {
    IPC  := fetch(PC)
    NNPC := exec(IPC)
    PC   := NPC
    NPC  := NNPC
}
-----------------------------------------
```

ARM etc.

```
-----------------------------------------
PC := 0x0
while (1) {
    IPC := fetch(PC)
    NPC := exec(IPC)
    PC  := NPC
}
```

Single core single thread AJIT processor core

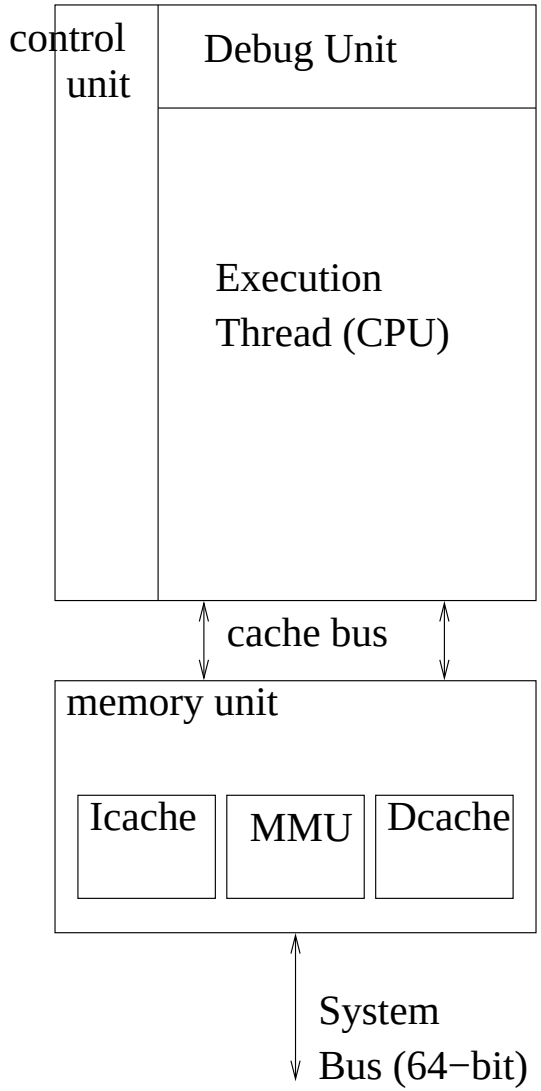


Figure: Single threaded core: 2.41 Coremarks/MHz, 2.37 DMIPS/Mhz, 0.4 WMIPS/MHz, 0.22 Linpack MFLOPS/MHz

AJIT processor pipeline constructed using AHIR-V2 tools

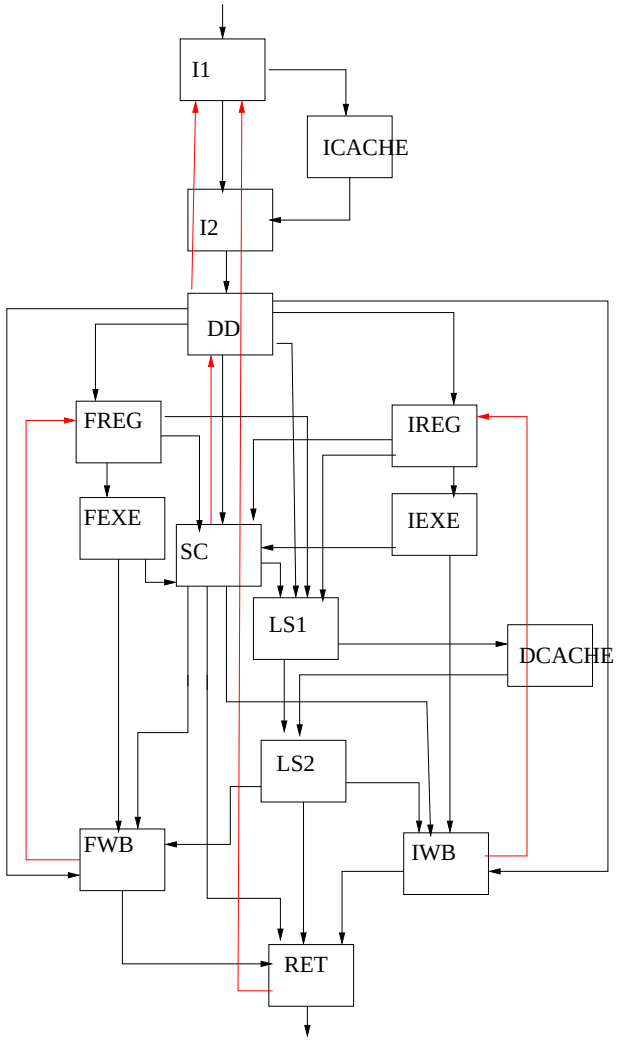


Figure: Case study: AJIT processor pipeline

Execution thread (CPU) characteristics

- IEEE 1754 standard ISA (Sparc V8).
- IEEE floating point unit.
- VIVT ICACHE (32kB) DCACHE (32kB) with 2-cycle latency on hit, MMU with 256 entry TLB and hardware page table walk.
- Single issue, in-order, seven stage one instruction-per-cycle instruction pipeline.
- Supports 64-bit arithmetic/logic/load/store instructions.
- Supports vector instructions (8/16/32-bit integer, half-precision/single-precision float).
- Integrated hardware debug unit, capable of full processor state visibility and controllability. Supports GDB remote debug.
- Execution pipeline with everything: 200K ASIC gates, 50K LUTs on FPGA.
- 2.41 Coremarks/MHz, 2.37 DMIPS/Mhz, 0.4 WMIPS/MHz, 0.22 Linpack MFLOPS/MHz.
- Boots Linux.

Execution thread: features

- Branch predictor (2-bit)
  - 256 entries (configurable).
- Return address stack
  - 16 entry (configurable).
- Partially out-of-order
  - FP/Integer/Load-store instruction execution can go on simultaneously.
- Debug support unit.
- Precise exceptions.

The AJIT processor development cycle using AHIR-V2

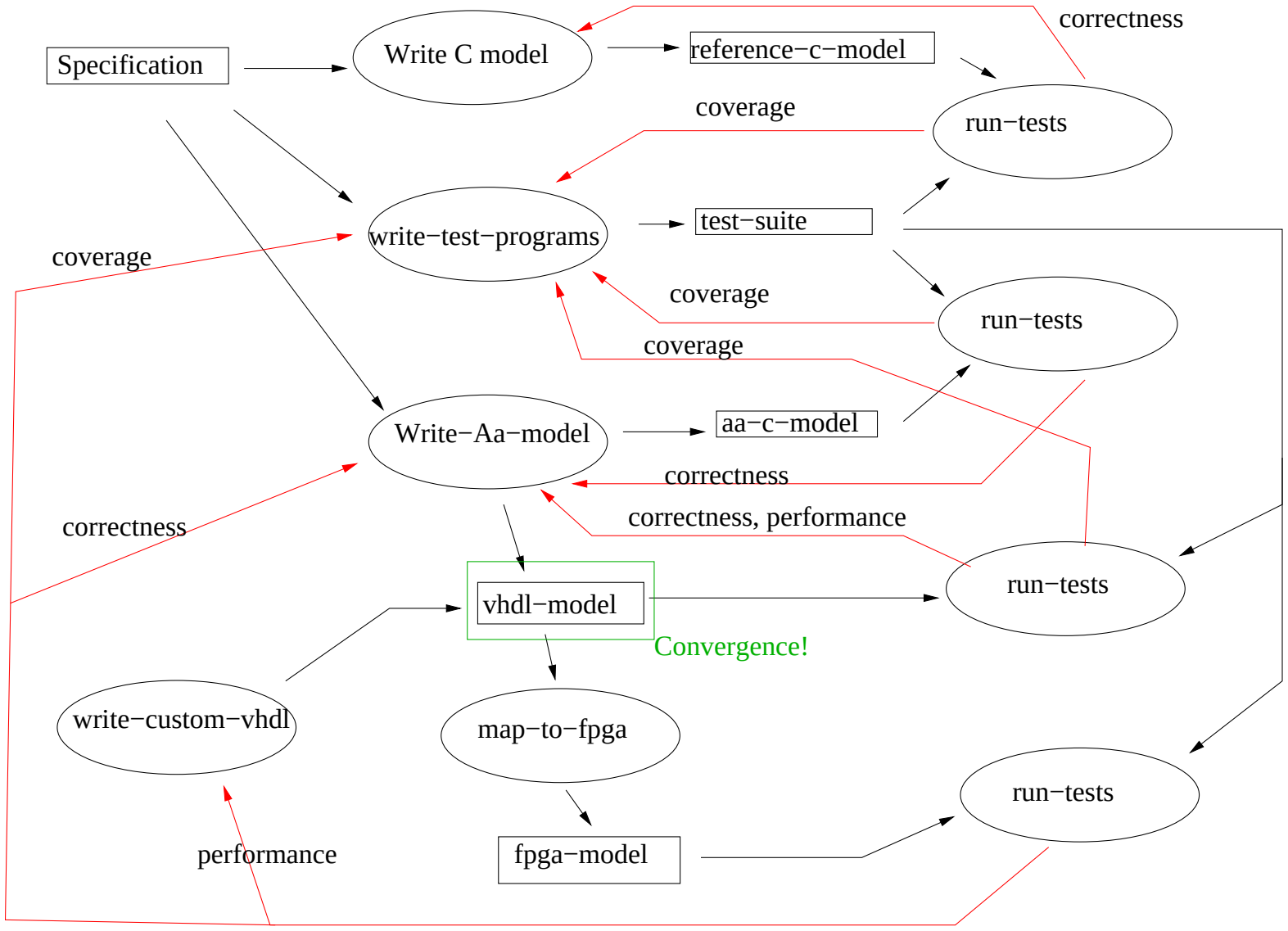
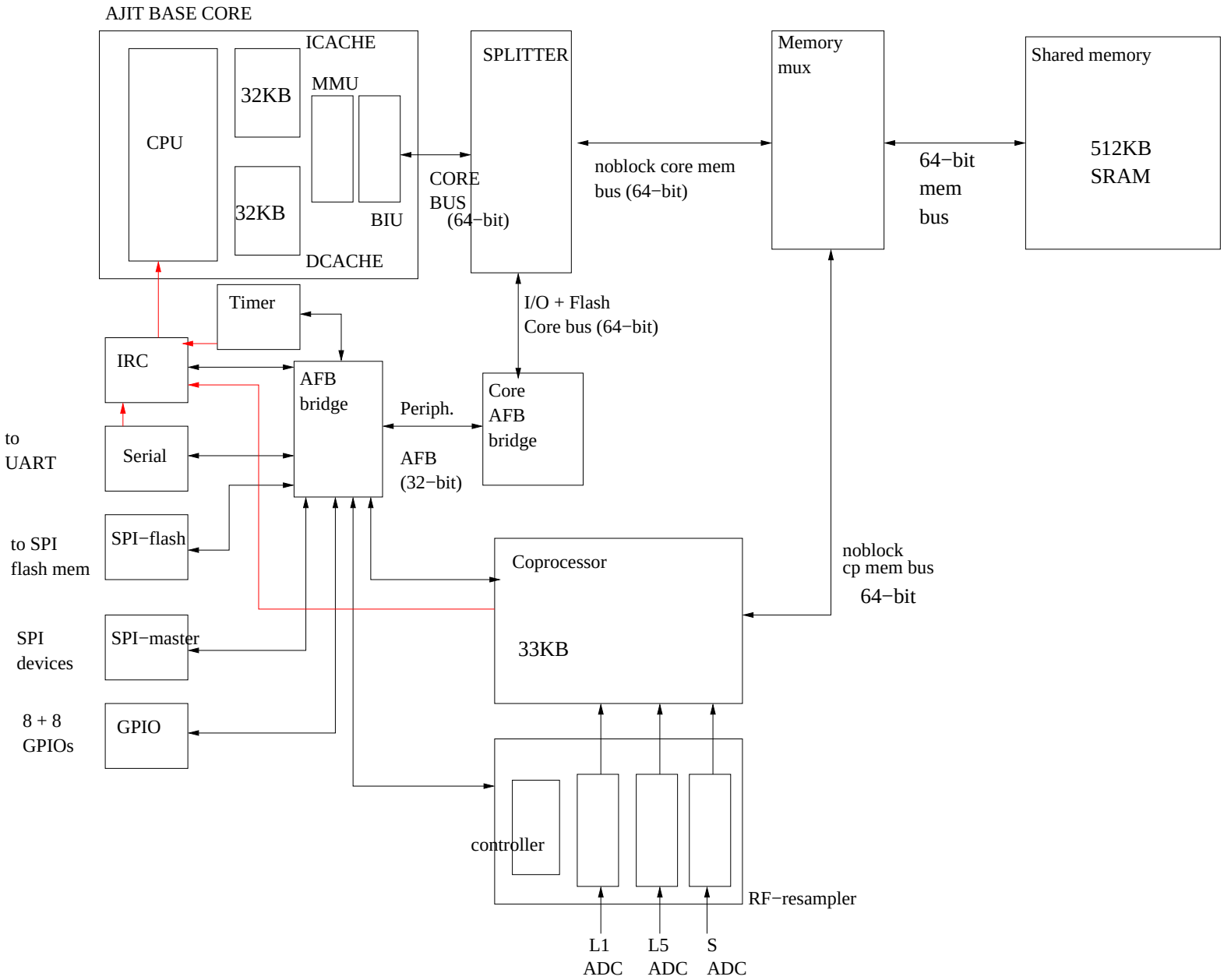


Figure: AJIT processor design flow

AJIT based IITB-NAVIC SOC



AJIT based IITB-NAVIC SOC

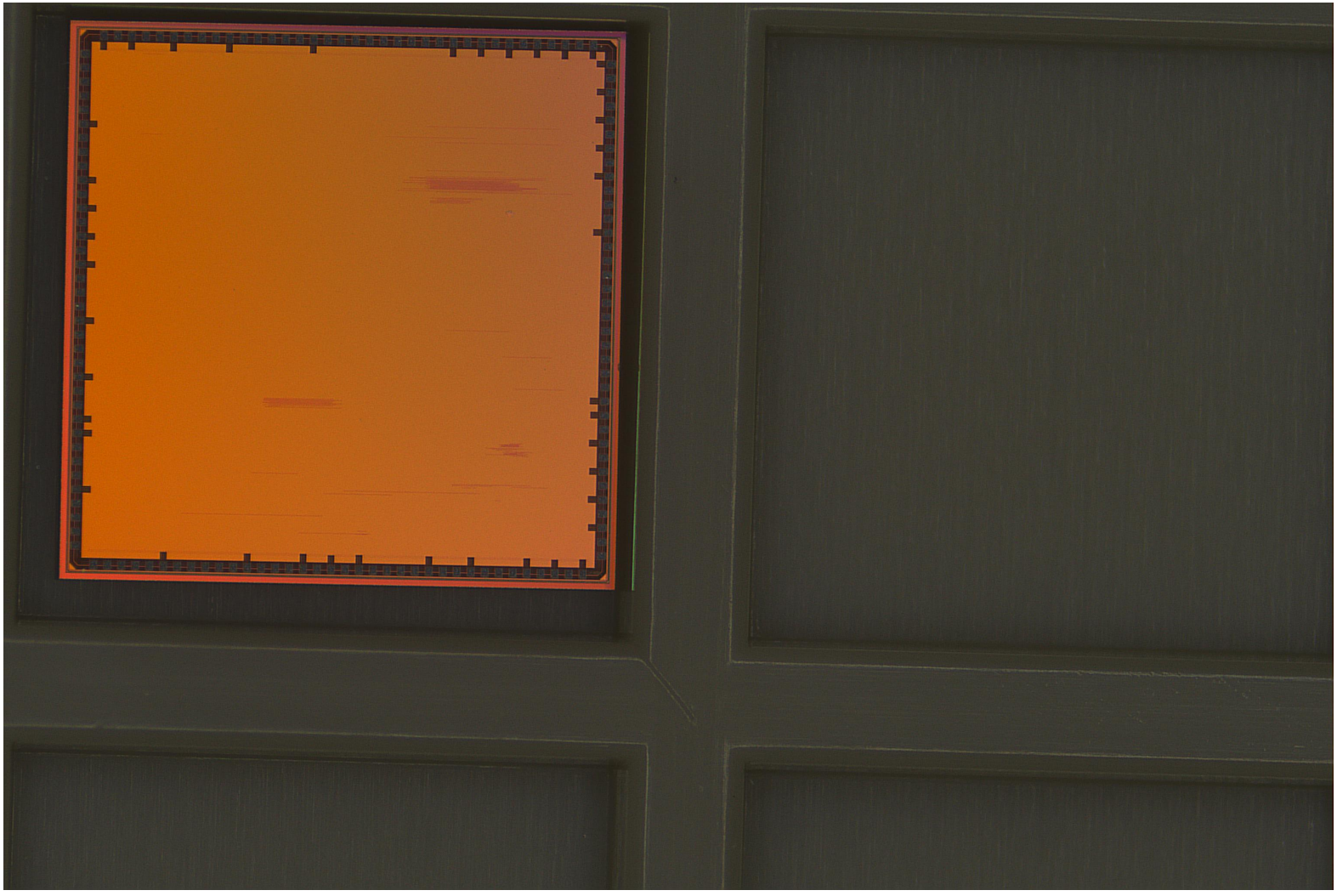


Figure: NAVIC base-band processing SOC die: 4mmX4mm, 1M gates, 512KB SRAM (single-cycle), 125MHz

Extensions to the single core single thread AJIT processor core

- ISA extensions: integer/float vector Extensions, 64-bit integer ALU, half-precision floating point support, compare and swap instruction.
- Dual-thread per core.
- Multi-core.
- Many-core (in progress).

Single core two thread AJIT processor core: double the performance with marginal increase in cost

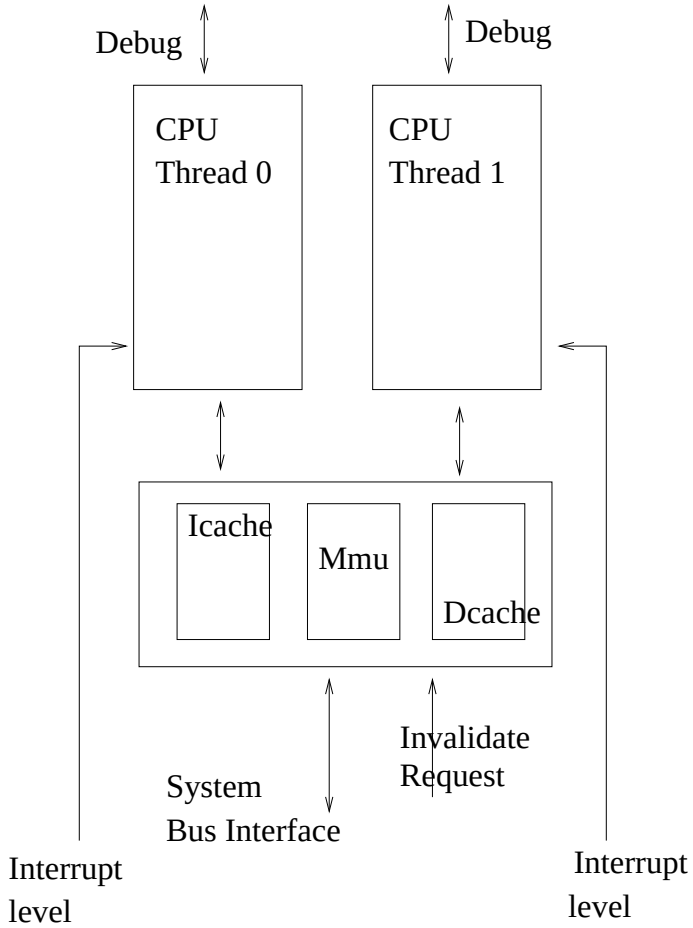


Figure: Dual threaded core: Up to 1.9X performance boost relative to single threaded core

Single core two thread AJIT processor core: double the performance with marginal increase in cost

task	speedup
Matrix-mul	1.88X
FFT	1.78X
ECG-processing	1.79X
Coremark	1.79X
Merge-sort	1.75X

4x2x64 multi-core: high performance embedded applications

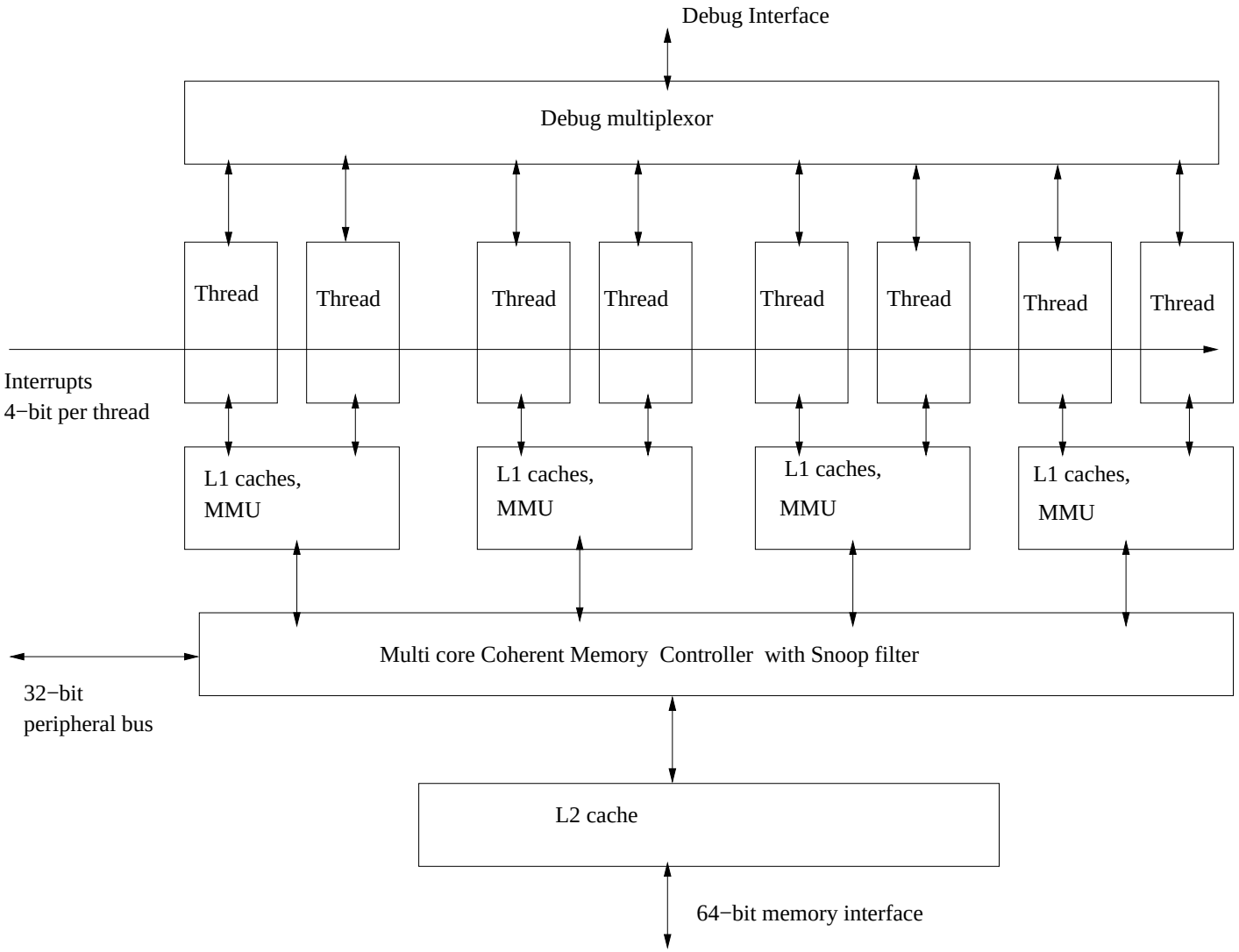


Figure: Quad-core eight-thread (4x2x64) processor

Multi-core implementation: key features

- Invalidate mechanism inside core caches.
- Coherent memory controller.
  - Snoop invalidate using write-through property of core caches, with novel snoop filter mechanism.
- Programmable interrupt controller.
  - Can handle 128 primary interrupt sources.
  - Complete flexibility in mapping interrupt sources to execution threads.
  - Inter-processor interrupt mechanism.
- L2 cache (8-way set associative write-back cache, up to 1MB).

Summary: AJIT processor configurations

ISA	32/64
NCORES	1 to 4
NTHREADS/CORE	1 or 2
FP-UNIT/THREAD	Present/Absent
L1 ICACHE/CORE	Size, associativity (write-through, VIVT)
	Present/Absent
L1 DCACHE/CORE	Size, associativity (write-through, VIVT)
	Present/Absent
MMU	TLB-size
	Present/Absent
L2 CACHE	Size, associativity (write-back, PIPT)
	Present/Absent

A very large portfolio of processors, catering to several application areas.

Attaching an AI accelerator

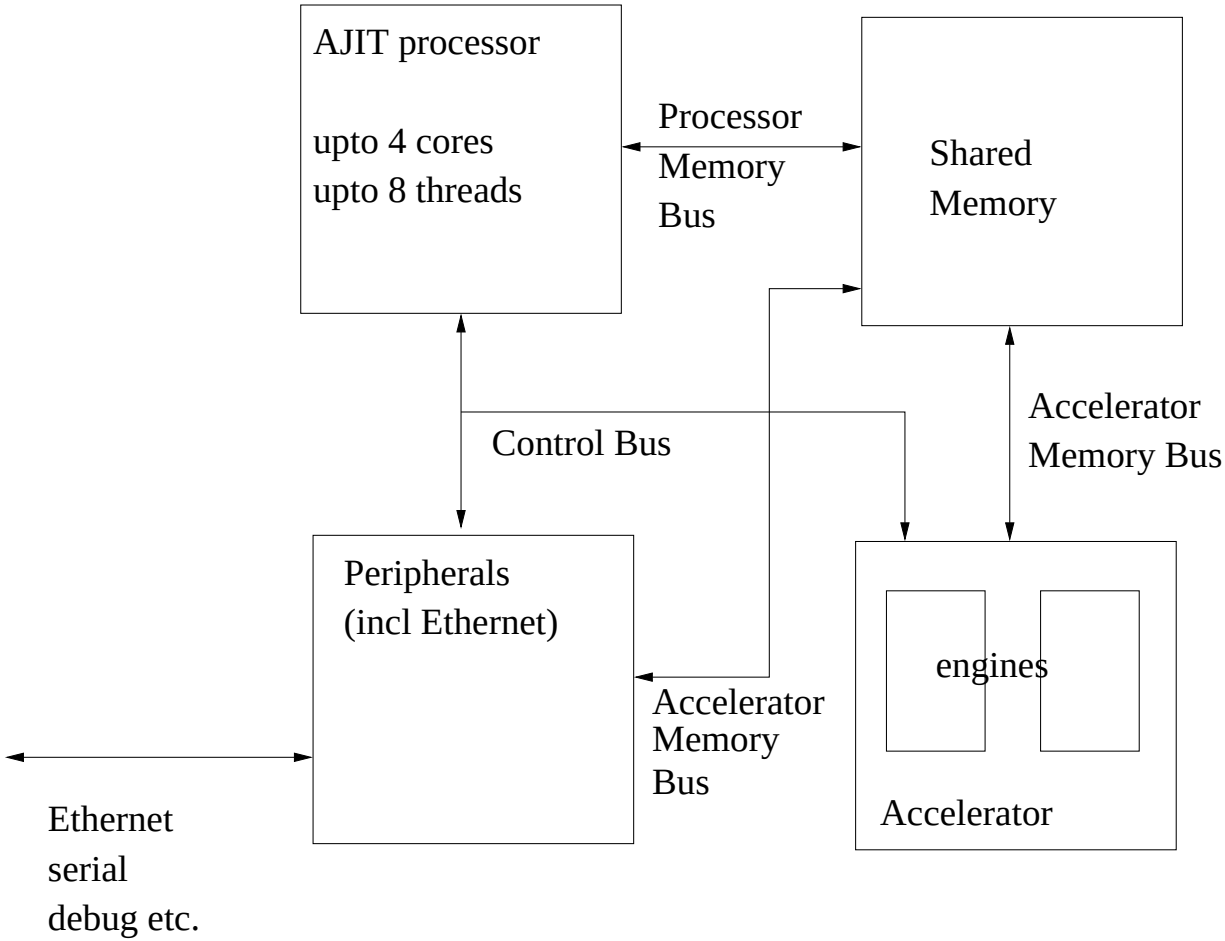


Figure: Visualization of SOC with AJIT and AI/ML accelerator

Proof of concept

- System on chip with single-core AJIT processor, AI/ML acceleration engines (1 to 8 parallel engines), Ethernet interface.
- Validated on FPGA.
- UNET convolution neural network implemented in SOC.
  - 85 GOPs per engine at 125MHz.
  - 8 parallel engines.
  - 5 221x221 images per second at 125MHz.

On-going: AJIT Many-Core

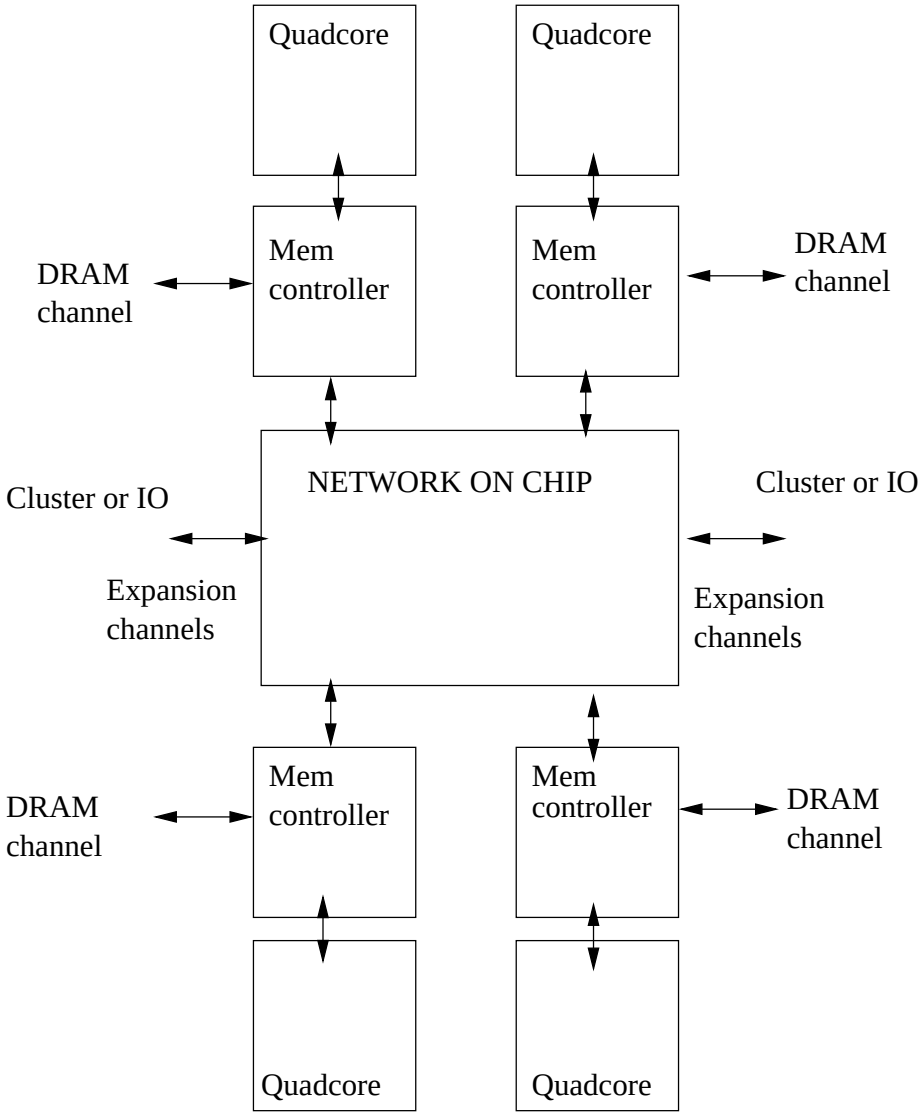


Figure: In progress: AJIT Many-core

Open-source AJIT tool-chain

- ssh://git@github.com/adhuliya/ajit-toolchain
- GCC/G++ Compiler, Assembler, Linker, Debugger, UCLIBC, GLIBC, Linux (3.16.1).
- Hardware access routines.
- Device access routines: serial, timer, interrupt controller, SPI, I2C, GPIO.
- ISR, generic trap handling.
- Cooperative RTOS (CORTOS2).
- Linux.

Teaching with the AJIT processor: wishlist

- Bare-metal
  - Initialization of run-time environment (processor state, stack, memory maps).
  - Critical devices: timers, interrupt controllers, UARTs, Ethernet, ADC/DAC, PWM etc.
  - Memory management and protection.
  - Interrupt and exception handlers.
- Embedded systems
  - Cooperative RTOS: Schedulers, semaphores, locks, queues, coroutines (light-weight threads) memory allocators.
  - Preemptive RTOS: Preemptive schedulers etc.
  - Multi-threading and parallel programming.
- System-on-chip design.
  - Adding accelerators (hardware, drivers).
- Projects
  - EDL, B.Tech., M.Tech., Ph.D.

Platform (in progress)

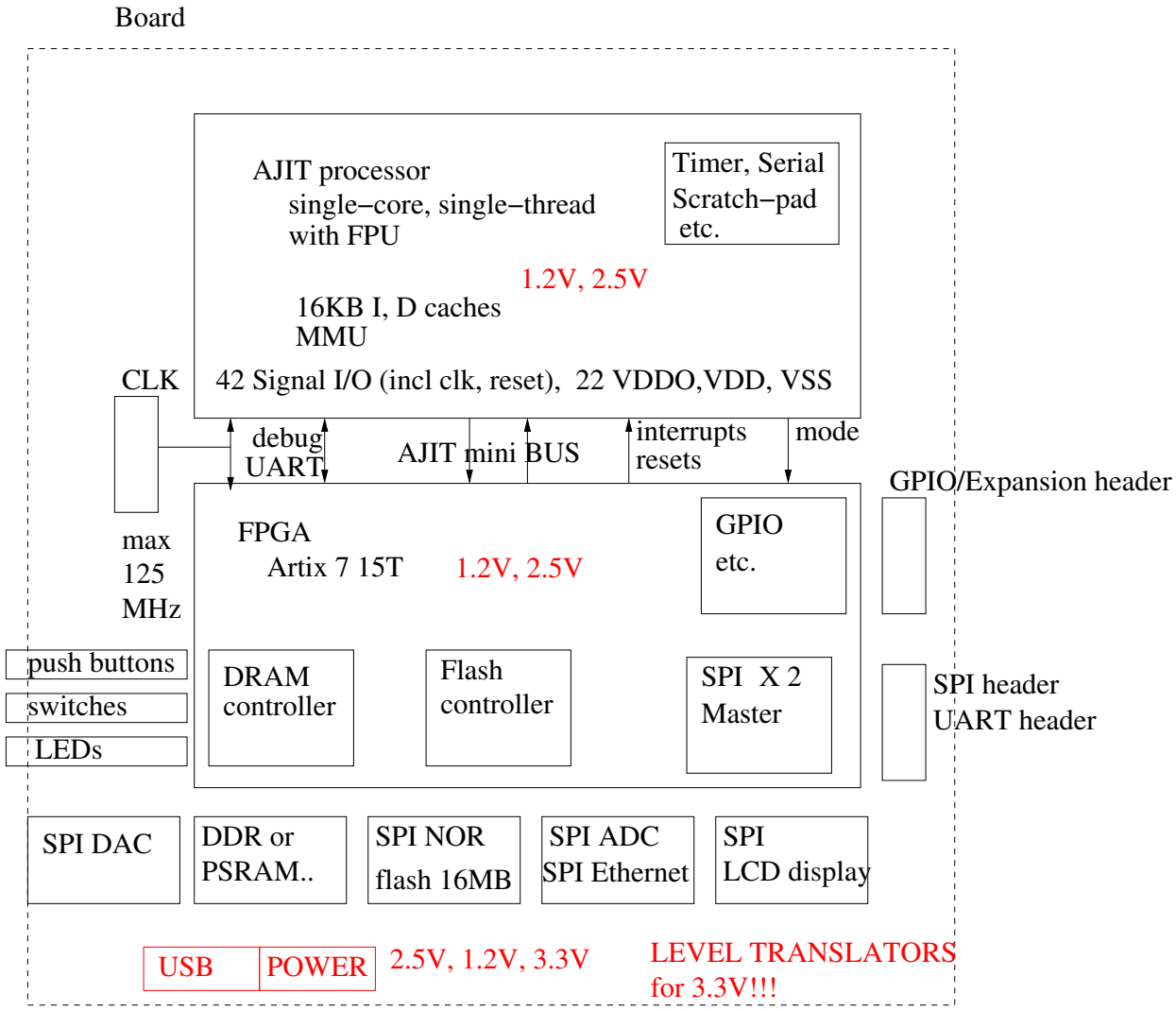


Figure: Platform version-0

Looking ahead

- Teach students to move hardware design to a higher-level
  - productivity.
  - faster verification flow.
    - Verify at higher level.
    - Correct by construction hardware generation.
  - learning curve!
- Dig deep into the micro-processor: under the hood and up the software hierarchy.
- Teach students hardware-software co-design concepts and patterns, all the way to actual hardware.
- Substantially improve quality of student work (learning/research/development/industry).