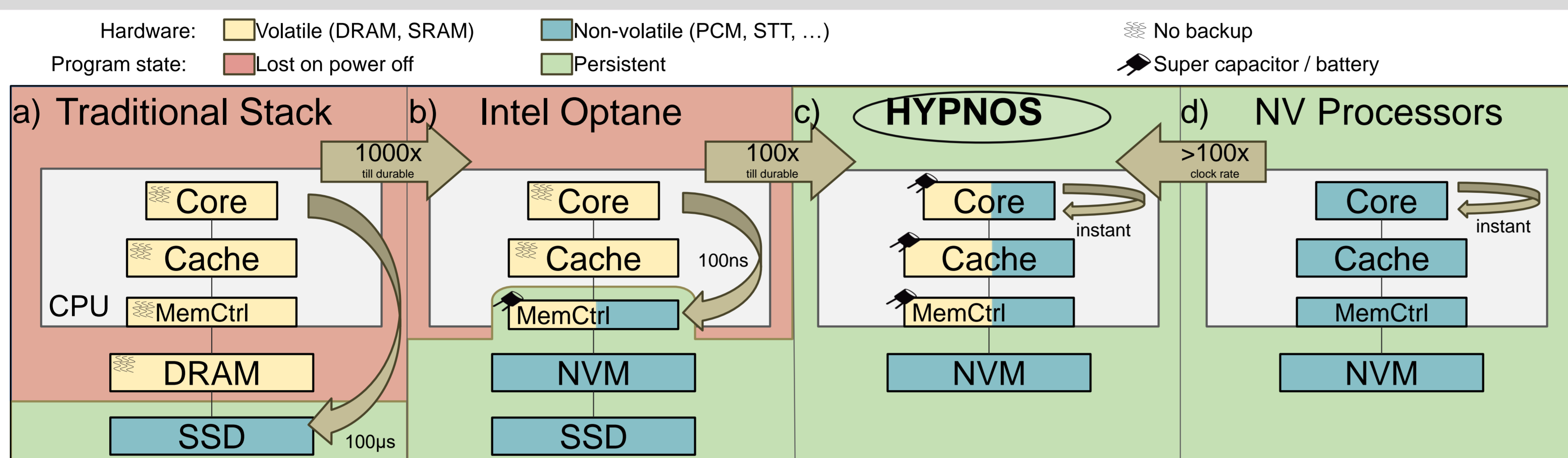


HYPNOS

Co-Design of Persistent, Energy-efficient and High-speed Embedded Processor Systems with Hybrid Volatility Memory Organisation

Prof. Dr.-Ing. Jürgen Teich, Dr.-Ing. Stefan Wildermann
Hardware/Software Co-Design, Friedrich-Alexander-Universität Erlangen-Nürnberg

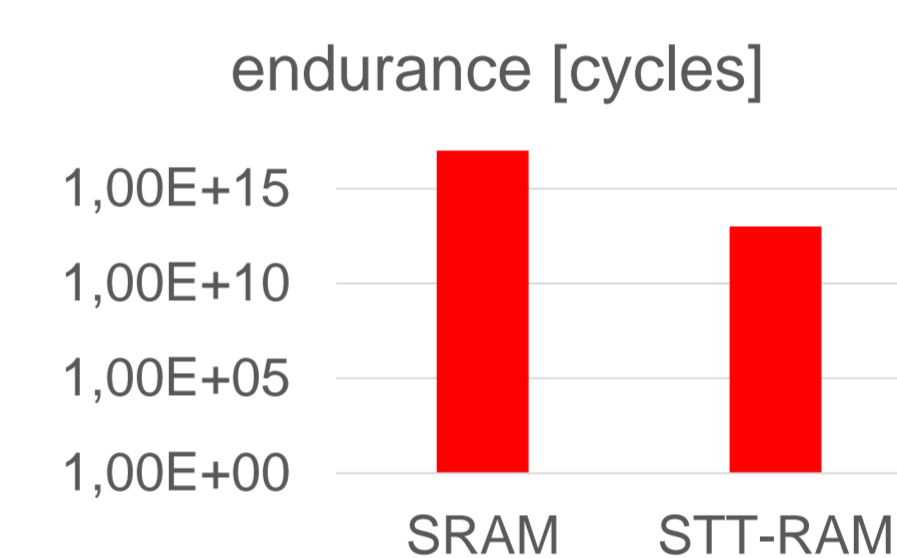
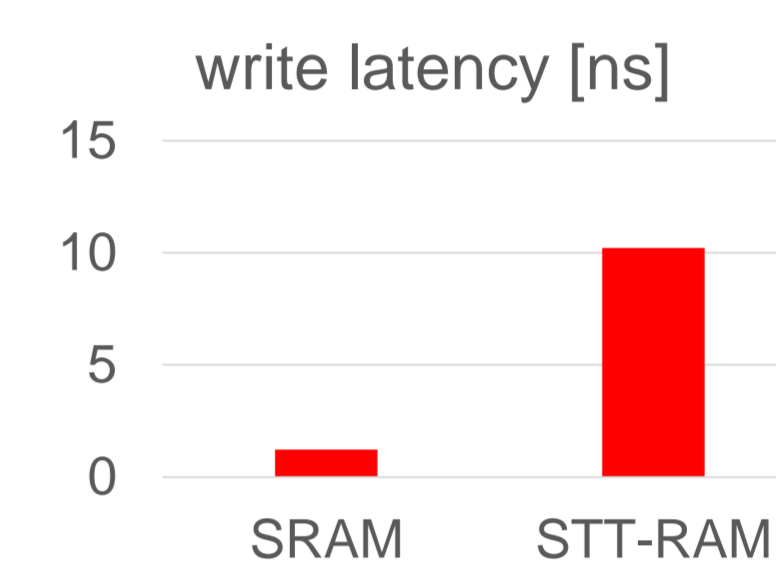
Overview: Computer System Architectures



Explicit block-grained persistence	Explicit cache line-grained persistence	Instruction & transaction-level persistence	Instruction-level persistence
<ul style="list-style-type: none"> +High CPU clock speeds -Explicit serialization -Very high latency for persistence -No persistent transactions <pre>void append(int fd, int ele) { int buf[PAGE_SIZE / 4]; pread(fd, buf, PAGE_SIZE, 0); int next = buf[0]; buf[0]++; buffer[next] = element; pwrite(fd, buf, PAGE_SIZE, 0); }</pre>	<ul style="list-style-type: none"> +High CPU clock speeds -Explicit ordering & flushes -High latency for persistence -No persistent transactions <pre>void append(int *buf, int ele) { int next = buf[0]; buf[next] = ele; clwb(&buf[next]); sfence(); buf[0]++; clwb(&buf[0]); sfence(); }</pre>	<ul style="list-style-type: none"> +High CPU clock speeds +Simple programming model +Low latency for persistence +Low energy +Persistent transactions <pre>void append(int *buf, int ele) { int next = buf[0]; buf[0]++; buf[next] = ele; }</pre>	<ul style="list-style-type: none"> -Low CPU clock speeds +Simple programming model +Low latency for persistence +Low energy -No transactions <pre>void append(int *buf, int ele) { int next = buf[0]; buf[0]++; buf[next] = ele; }</pre>

The Gaps

- Performance:** High write latency of NVM technology
- Endurance:** At least four orders of magnitude lower endurance than SRAM
- Programmability:** Implementing persistent data structures is difficult



```
void append(int *buf, int ele) {
    int next = buf[0];
    buf[next] = ele;
    clwb(&buf[next]);
    sfence();
    buf[0]++;
    clwb(&buf[0]);
    sfence();
}

void append(int *buf, int ele) {
    int next = buf[0];
    buf[0]++;
    buf[next] = ele;
}
```

Working Plan and Methods

WP1	WP2	WP3	WP4	WP5
Requirement Analysis <ul style="list-style-type: none"> Evaluate <i>benchmarks</i> ranging from IoT to database domains. Identify relevant <i>data structures</i> and their persistency requirements. Evaluate <i>characteristics of NVM technologies</i> and <i>tools</i> for simulation/analysis (like NVSIM, NVMain). 	Environment for CPU Architecture Design & Evaluation <ul style="list-style-type: none"> Use <i>gem5</i> to simulate different memory extensions and CPU controllers as investigated in other work packages. Integrate <i>NVM simulation</i> environments. Exploit <i>ARM's transactional memory extensions (TME)</i>. 	Design space exploration (DSE) of hybrid volatile memory hierarchy CPU architectures <ul style="list-style-type: none"> <i>Architecture exploration</i> of memory options including i) volatile vs. non-volatile vs. hybrid, ii) memory technology selection, iii) dimensioning, iv) memory extensions <i>DSE with multiple objectives:</i> i) parametric hardware (design) cost, ii) execution time, iii) energy consumption and iv) endurance. 	Checkpoint and Recovery Control Architecture Design <ul style="list-style-type: none"> Support for <i>instruction-level persistence</i>: Upon a power failure, persist the state of computation. After the power is back, the program continues its execution. Support for <i>transaction-level persistence</i>: Upon power failure persist pre-transactional values. Properly dimensioned battery/supercapacitors to support checkpointing. 	Programming Support <ul style="list-style-type: none"> Investigate <i>programming techniques</i> for persistent data management. <i>Hardware-supported transaction</i> for transaction-level persistence. <pre>status = __tstart(); if (status == 0) { // transaction code // ... __tcommit(); }</pre> Transactional memory extensions

