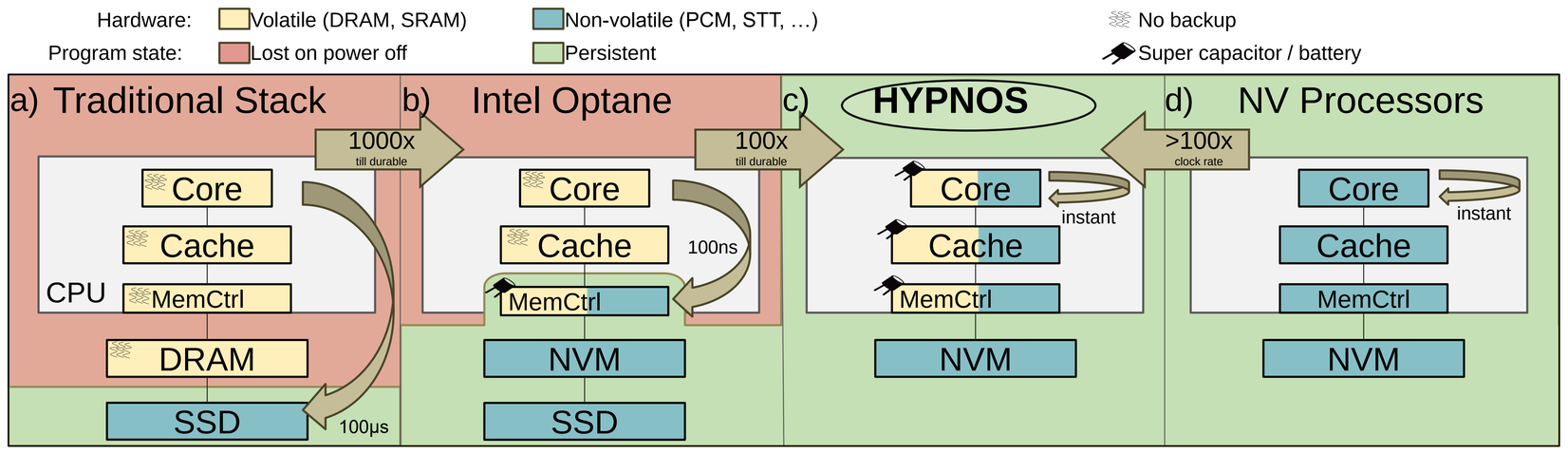
Friedrich-Alexander-Universität
Technische Fakultät

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

+High CPU clock speeds
-Explicit serialization
-Very high latency for persistence
-No persistent transactions

```
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);
}
```

Explicit cache line-grained persistence

+High CPU clock speeds
-Explicit ordering & flushes
-High latency for persistence
-No persistent transactions

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

Instruction & transaction-level persistence

+High CPU clock speeds
+Simple programming model
+Low latency for persistence
+Low energy
+Persistent transactions

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

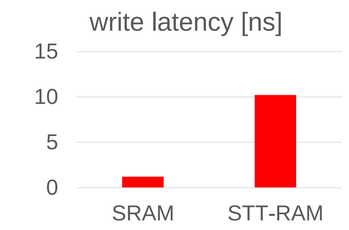
Instruction-level persistence

-Low CPU clock speeds
+Simple programming model
+Low latency for persistence
+Low energy
-No transactions

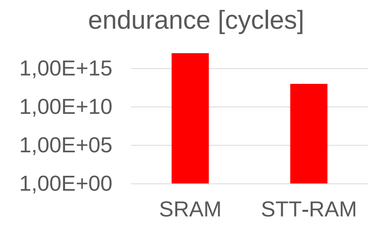
```
void append(int *buf, int ele) {
    int next = buf[0];
    buf[0]++;
    buf[next] = ele;
}
```

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

Requirement Analysis

- Evaluate *benchmarks* ranging from IoT to database domains.
- Identify relevant *data structures* and their persistency requirements.
- Evaluate *characteristics of NVM technologies* and *tools* for simulation/analysis (like NVSIM, NVMain).

WP2

Environment for CPU Architecture Design & Evaluation

- Use *gem5* to simulate different memory extensions and CPU controllers as investigated in other work packages.
- Integrate *NVM simulation* environments.
- Exploit *ARM's transactional memory extensions* (TME).

gem5

WP3

Design space exploration (DSE) of hybrid volatile memory hierarchy CPU architectures

- *Architecture exploration* of memory options including i) volatile vs. non-volatile vs. hybrid, ii) memory technology selection, iii) dimensioning, iv) memory extensions
- *DSE with multiple objectives*: i) parametric hardware (design) cost, ii) execution time, iii) energy consumption and iv) endurance.

WP4

Checkpoint and Recovery Control Architecture Design

- Support for *instruction-level persistence*:
  - Upon a power failure, persist the state of computation. After the power is back, the program continues its execution.
- Support for *transaction-level persistence*: Upon power failure persist pre-transactional values.
- Properly dimensioned battery/supercapacitors to support checkpointing.

WP5

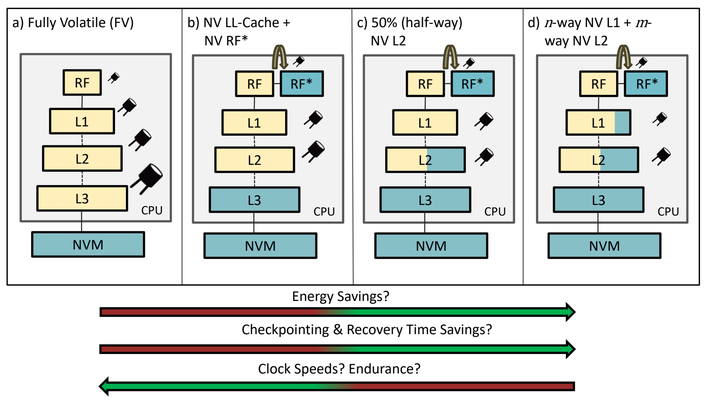
Programming Support

- Investigate *programming techniques* for persistent data management.
- *Hardware-supported transaction* for transaction-level persistence.

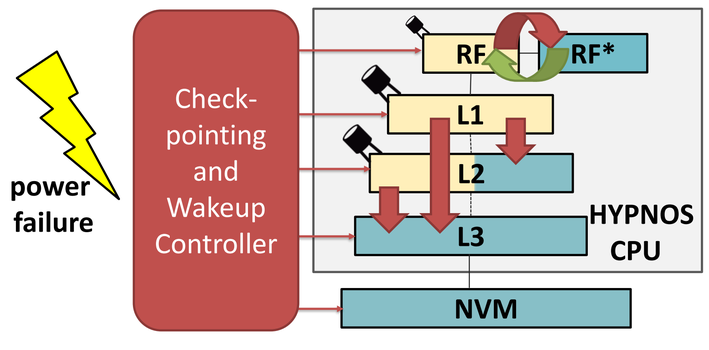
```
status = __tstart();
if (status == 0) {
  // transaction code
  // ...
  __tcommit();
}
```

Transactional memory extensions

Memory Architecture DSE



Checkpointing and Wakeup Controller



Hardware Transaction Memory