



Approximate Computing with Unreliable Memory: FPGA-Based Emulation for PULP

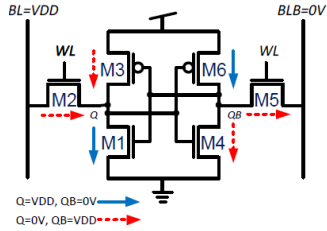
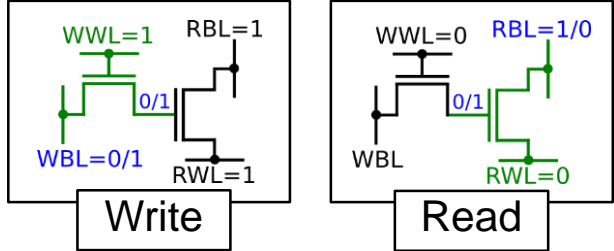
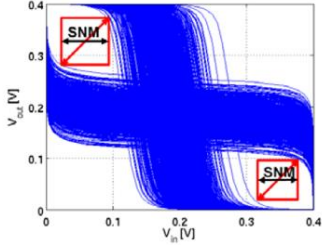
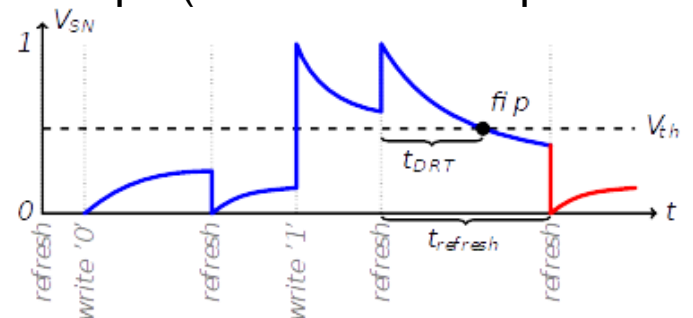
Marco Widmer, Andreas Burg

École Polytechnique Fédérale de Lausanne

Sept. 4, 2019

EPFL

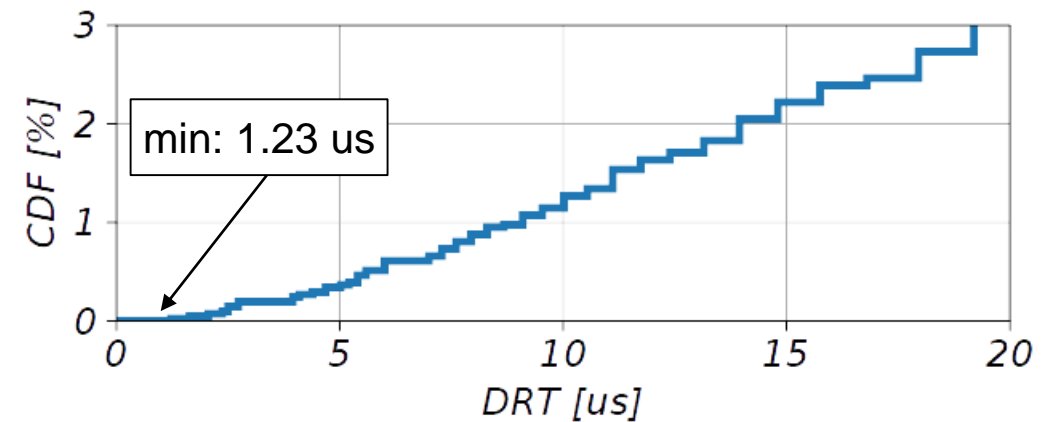
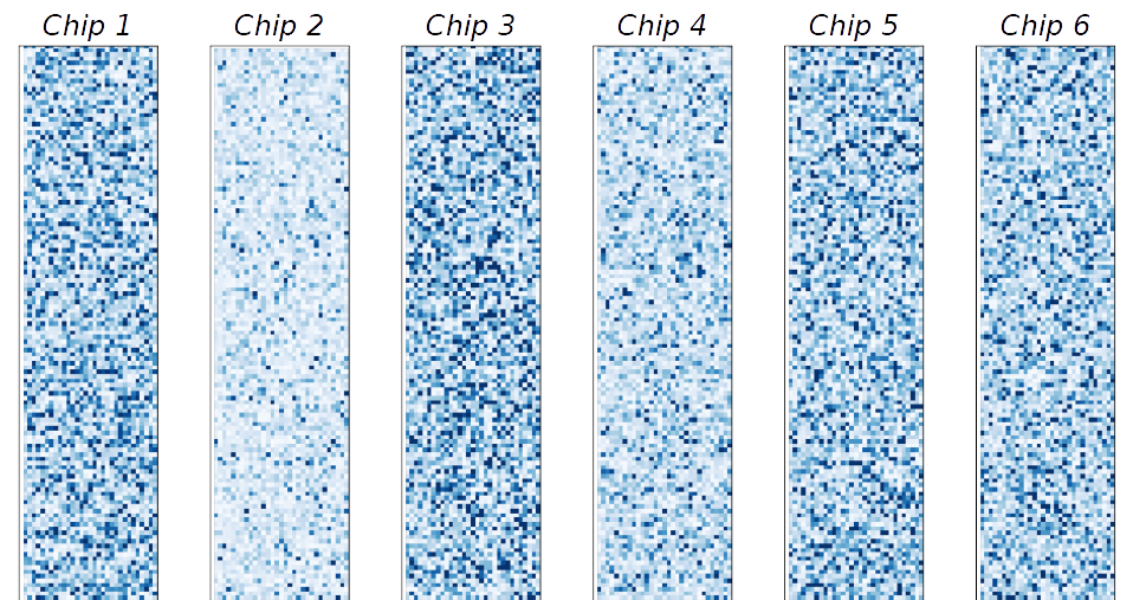
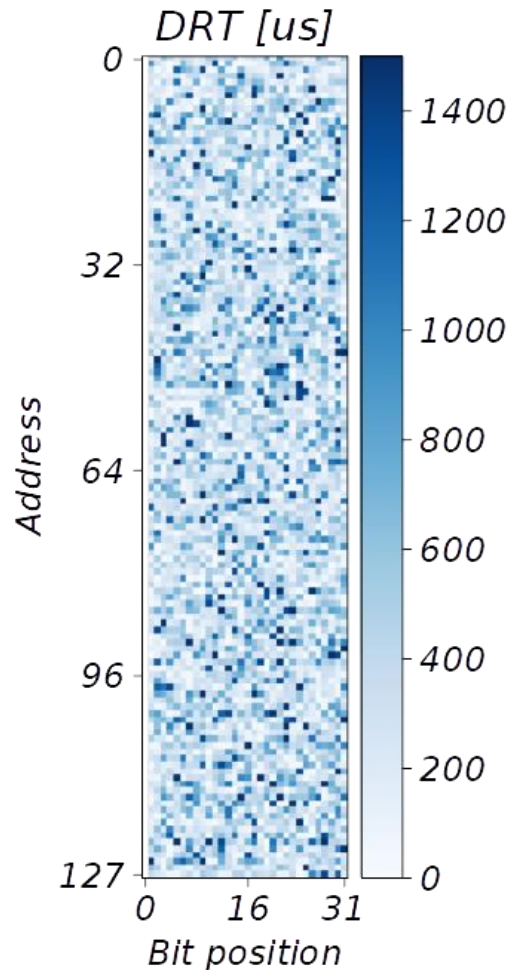
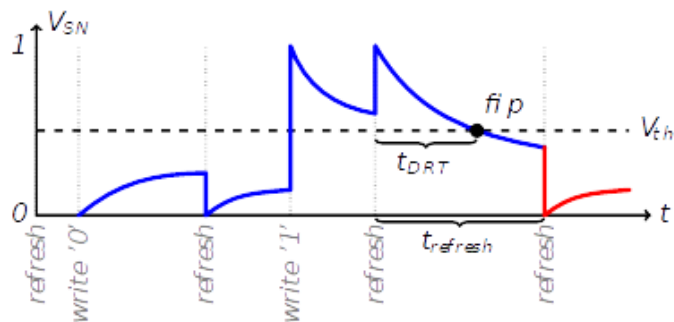
Reliability of Ultra Low Power On-Chip Memories

	SRAM	GC-eDRAM
Bit Cell		
Power saving technique	Voltage scaling	Refresh rate relaxation
Problem	Faulty cells 	Bit flips (data lifetime dependant) 

Power saving measures may cause faults in stored data

eDRAM Fault Frequency & Location

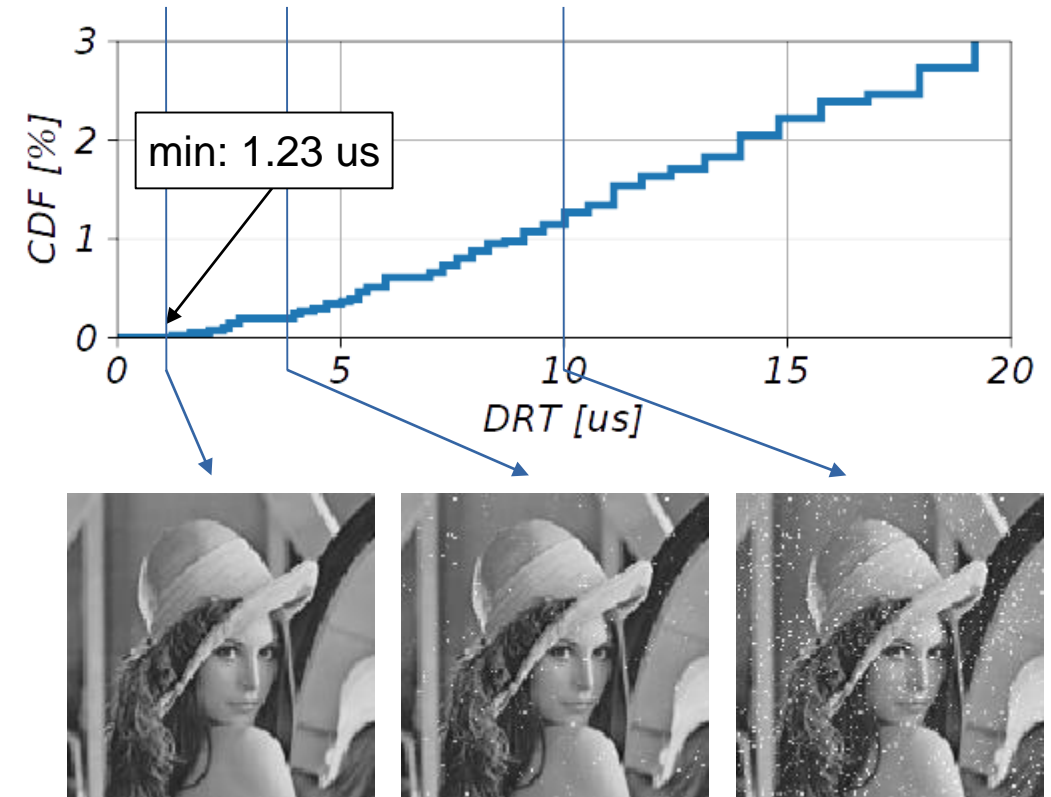
- Cell data retention time (DRT) depends on process corner, environmental conditions, local process variations
- If $DRT < \text{refresh interval}$: Bit flip
- For a given die: Low-DRT cells can be mapped
- DRT map varies with **die-to-die variations**



Data retention time (DRT) map of a gain-cell eDRAM in 28nm bulk.

Impact of Faults on Applications

- Impact depends greatly on the data stored at the fault site
- Faults in critical data (code, loop counters): Catastrophic
- Faults in non-critical data (image pixels, NN weights):
 - Depends on data representation
 - Faulty LSB: Small error, minor consequence on output
 - Faulty MSB: Large error
 - Faulty FP exponent: Extreme error
 - Depends on data lifetime

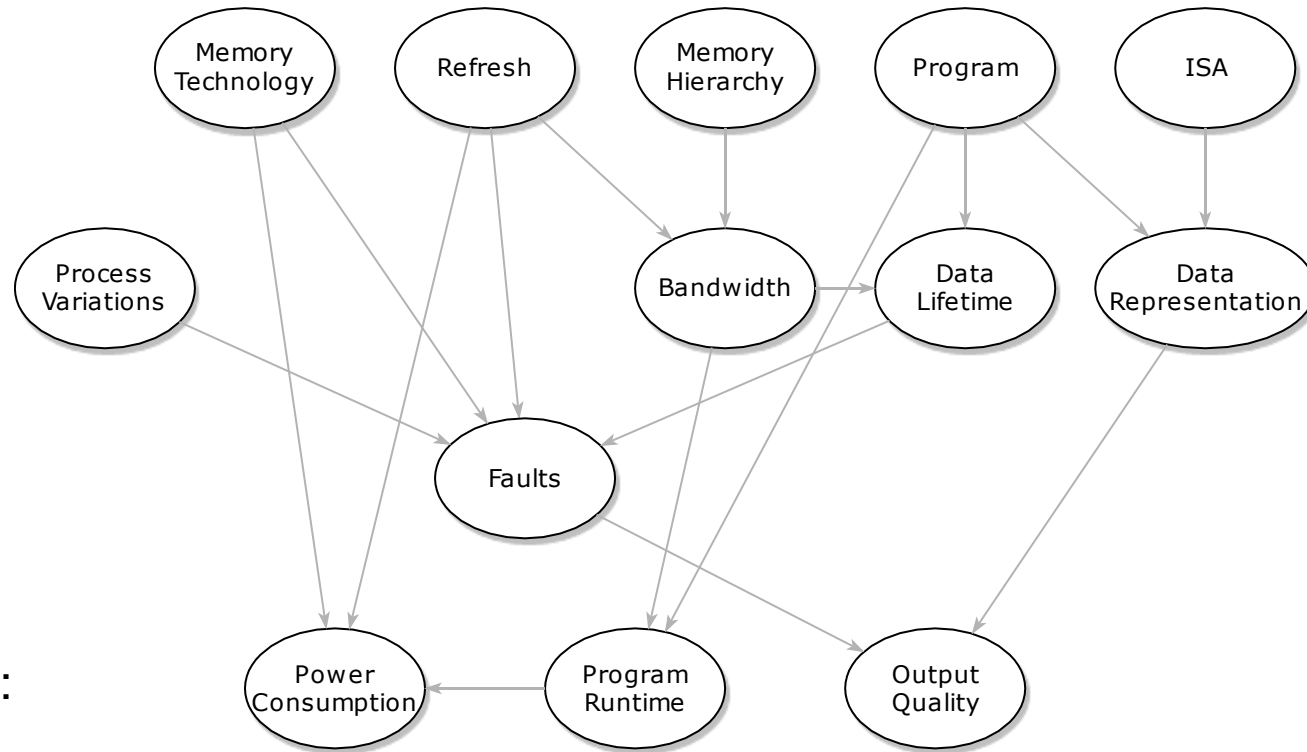


For a given application, there are good dies and bad dies.

How many dies give usable results? A Monte Carlo yield analysis is needed.

Memory System Complexity

Design Variables:

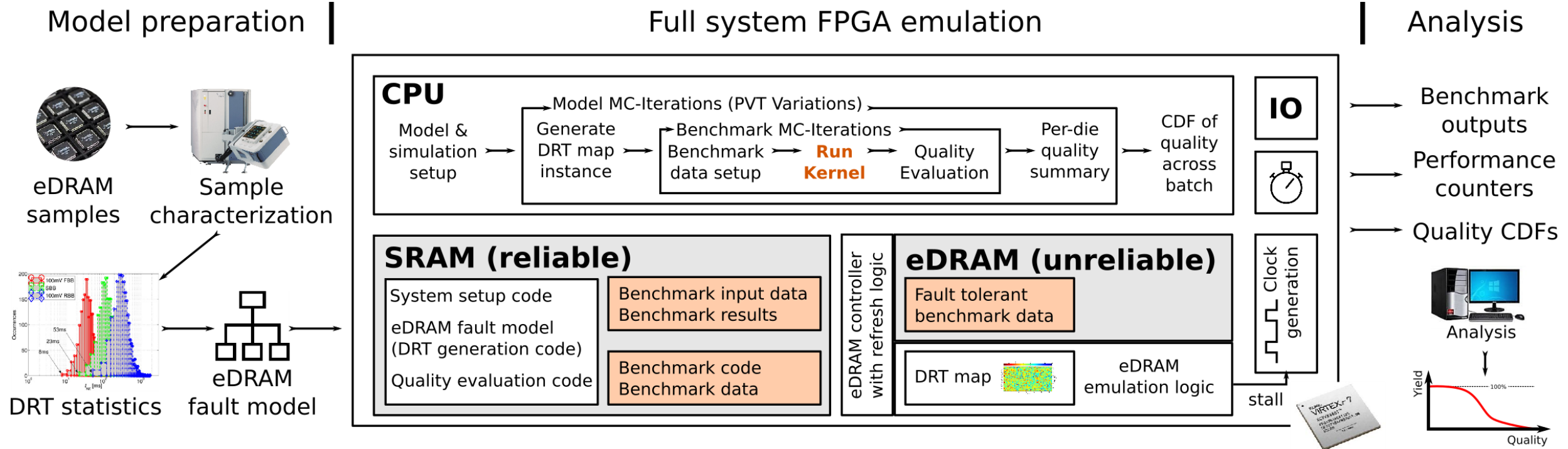


Consequences:

**Large design space, complex interactions.
Analytical assessment impossible.**

Solution: Yield analysis through emulation.

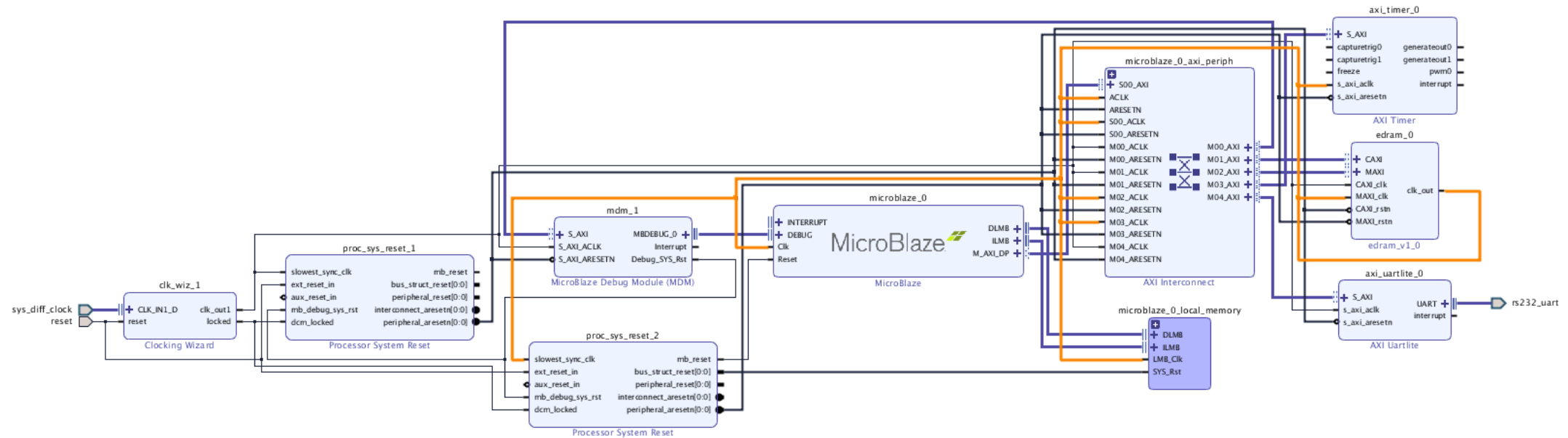
Previous Work: FPGA-Based eDRAM Emulator



- 1) A **fault model** is extracted from sample chips
- 2) CPU generates **random DRT maps** based on the fault model and initializes the eDRAM emulator
- 3) CPU runs a **benchmark** storing non-critical data in the **unreliable emulated eDRAM**
- 4) CPU analyzes the benchmark **output quality** and sends it to a computer for further analysis

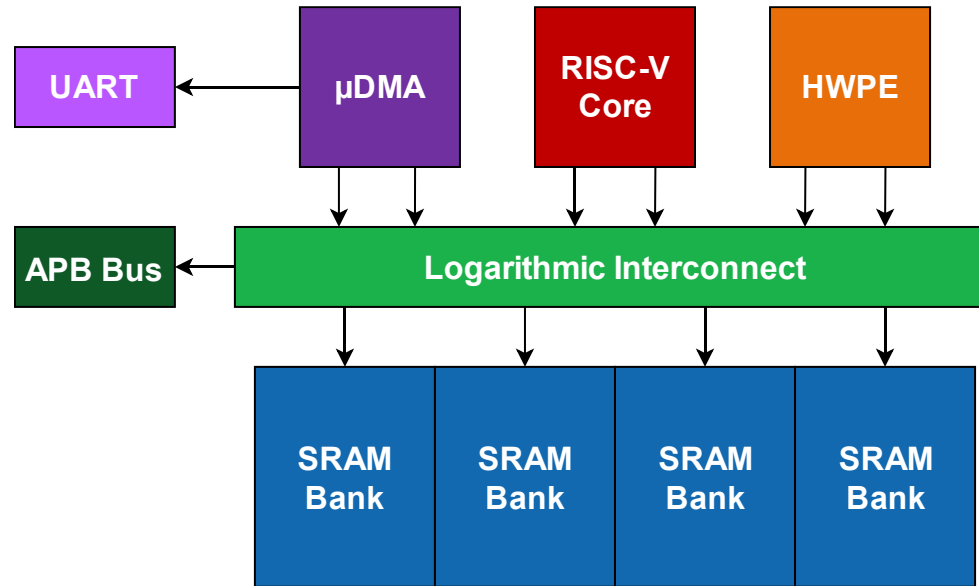
Goals of the Summer Project

- Until now: Xilinx MicroBlaze processing system
 - Slow memory access (AXI bus)



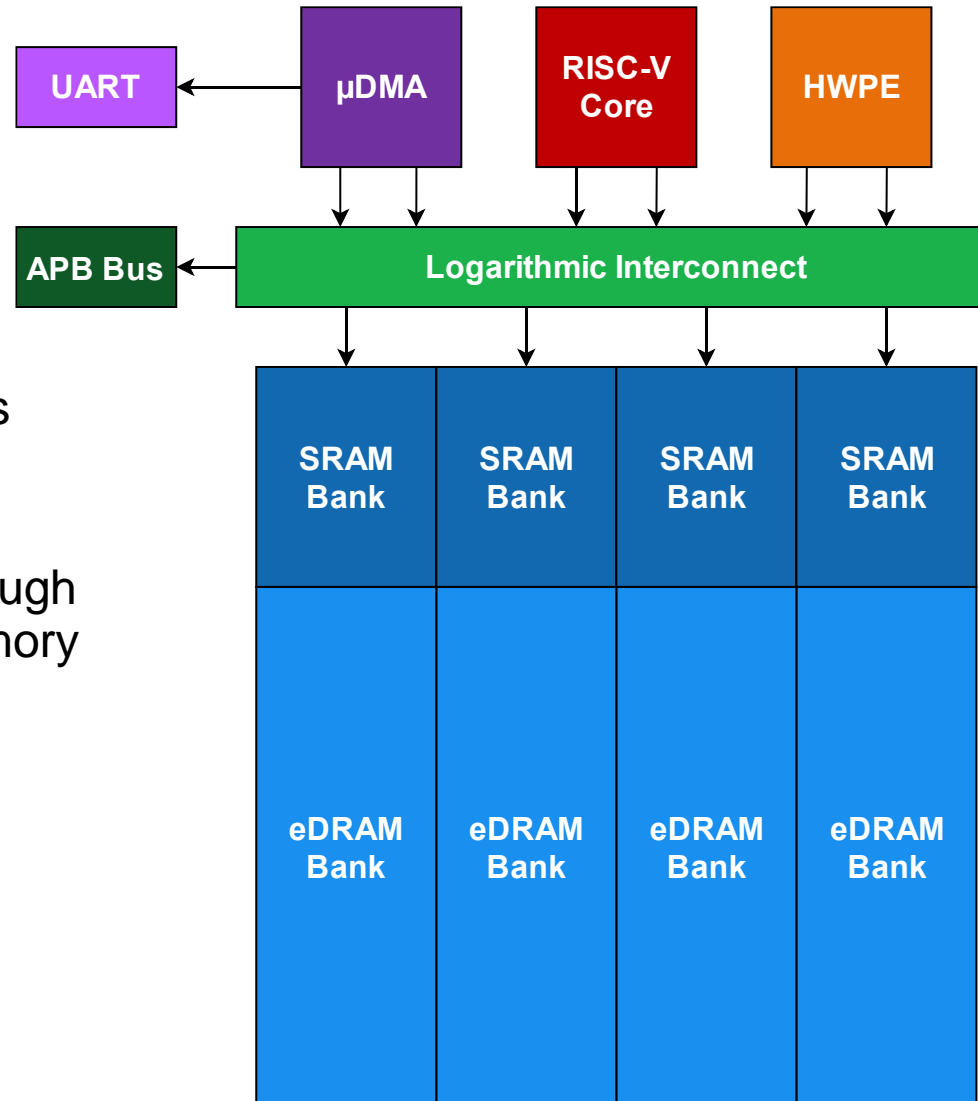
- Goal: Integrate emulator into the **PULP platform**

PULPissimo



- Pulpissimo: Single RISC-V core microcontroller
- FPGA port available
- Ported to Xilinx ZCU104 board during this project (upstreamed)

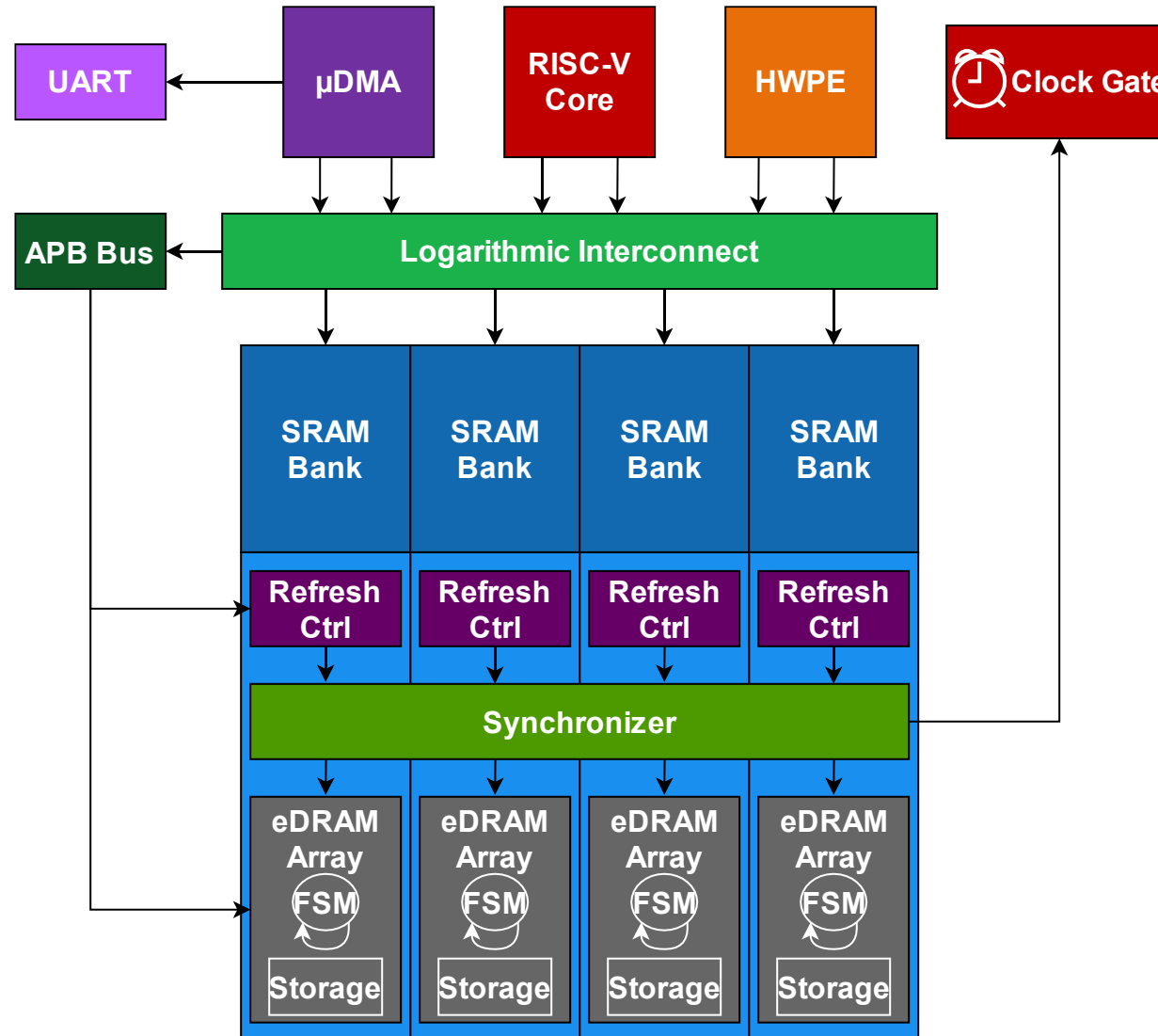
Emulator Integration into PULPissimo



- Interleaved SRAM banks extended with emulated eDRAM
- Single-cycle access through tightly coupled data memory (TCDM) interconnect

Emulator Integration into PULPissimo

- Emulator programmable through APB bus
- FSMs to model eDRAM behavior
- Clock gating to mask emulator overhead and retain cycle accuracy

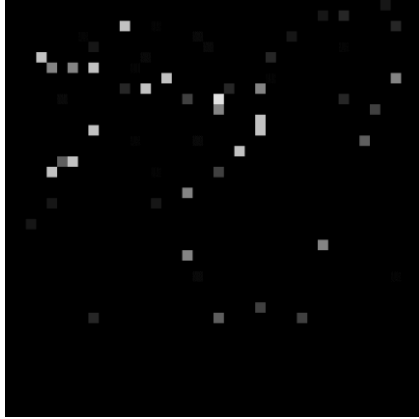


Implementation Figures

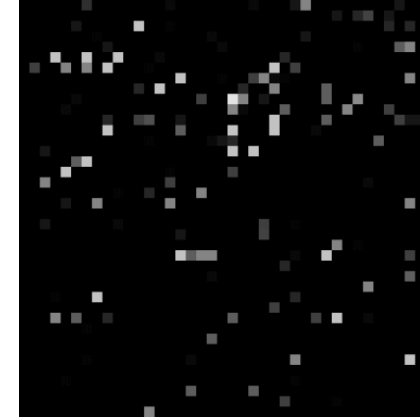


- SRAM: 1024 KiB
- eDRAM: 512 KiB
- FPGA runs at 20 MHz
- Emulation overhead:
 - Write access: no overhead
 - Read access: ~4x slower
 - Refresh cycle: ~100x slower (array refresh sequentialized)
 - 1% - 25% of all cycles, depending on refresh rate
 - Could be optimized

Software Example



```
events_init_rand(MU, SIGMA, MAX_DRT,  
    ACTION_DRT0, SEED);  
SET_RFINT(REFRESH_INTERVAL);  
EDRAM_START();  
  
char *image = malloc_edram(SIZE * SIZE);  
for (int i = 0; i < SIZE * SIZE; i++)  
    image[i] = 0;  
  
EDRAM_STOP();  
print_image(image);
```



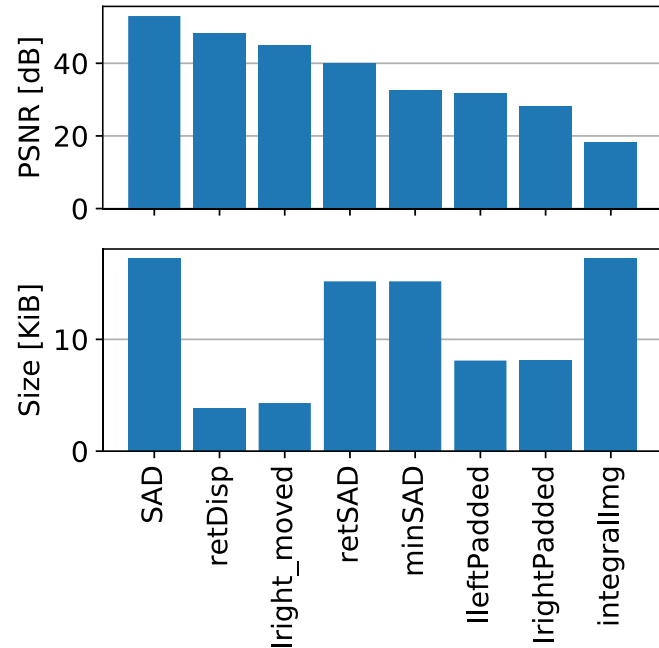
```
events_init_rand(MU, SIGMA, MAX_DRT,  
    ACTION_DRT0, SEED);  
SET_RFINT(REFRESH_INTERVAL);  
EDRAM_START();  
  
char *image = malloc_edram(SIZE * SIZE);  
for (int i = 0; i < SIZE * SIZE; i++)  
    image[i] = 0;  
sleep(100000);  
  
EDRAM_STOP();  
print_image(image);
```

Available Benchmarks

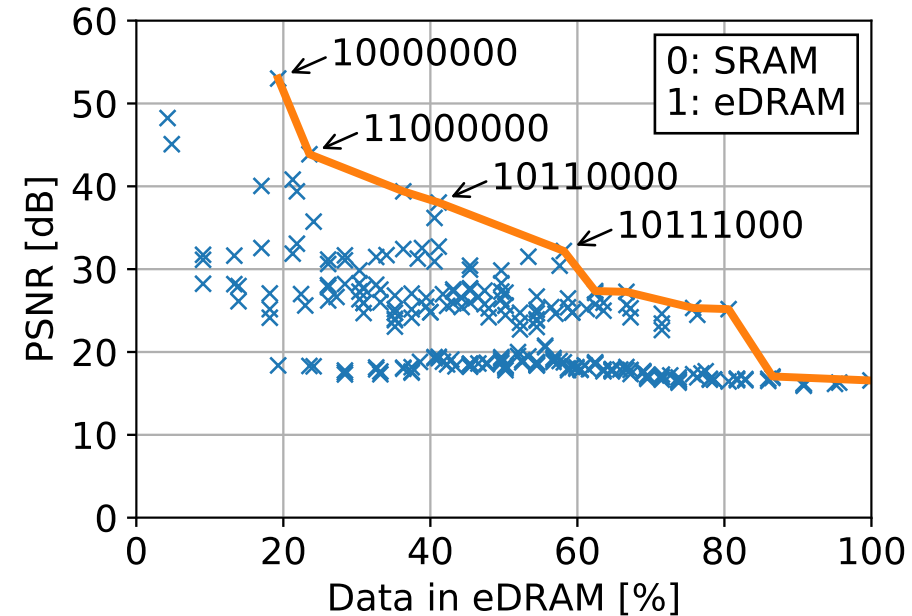
Benchmark	Description	Platforms	Remarks
Convolution	Edge detection	Xilinx, PULP	
Disparity	Stereo vision	Xilinx, PULP	
Susan	Image Smoothing	Xilinx, PULP	
MNIST	Neural network	Xilinx, PULP	
BNN	Binary neural network	PULP	not enough input vectors for quality analysis

Results: Relevance of Data Organization

- Data structures can be placed in **reliable** or **unreliable memory**
- Some data structures are more **error resilient** than others
- Our emulator allows exploration of different **data placement** schemes



Single data structure placed in unreliable memory

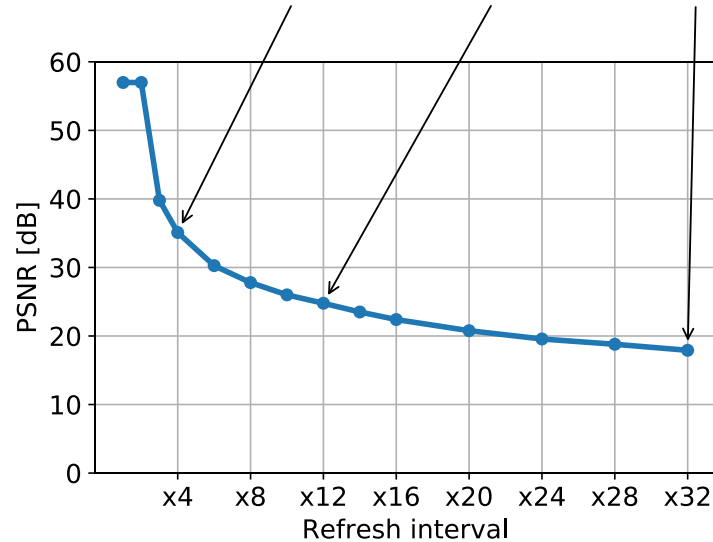
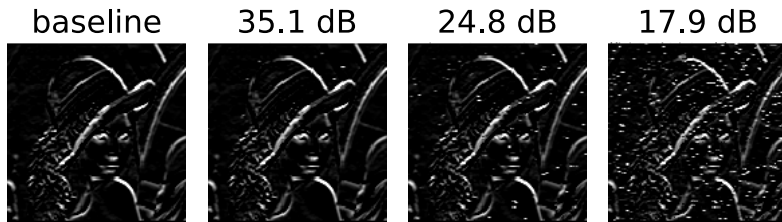


Combinations of data structures placed in unreliable memory with Pareto front

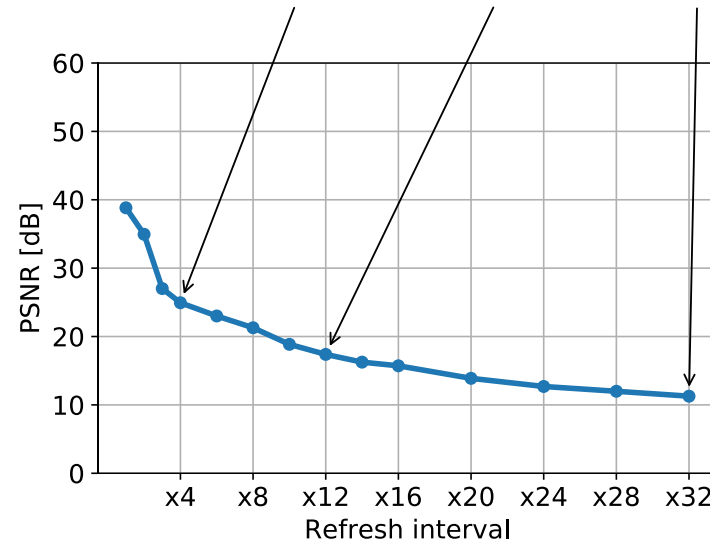
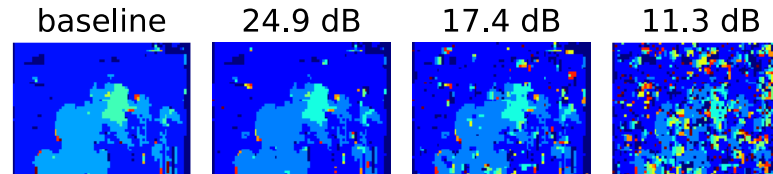
Careful selection of data to be placed in unreliable memory has a considerable quality impact.

Results: Benchmark Execution at Sub-Critical Refresh Rates

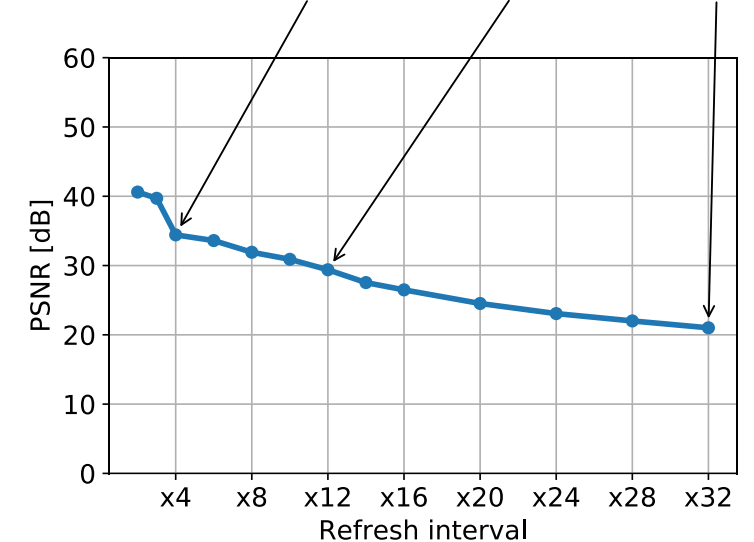
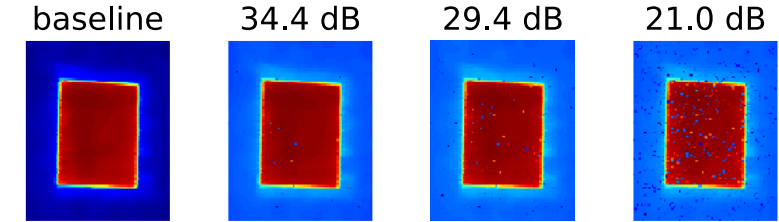
Convolution



Disparity



Susan

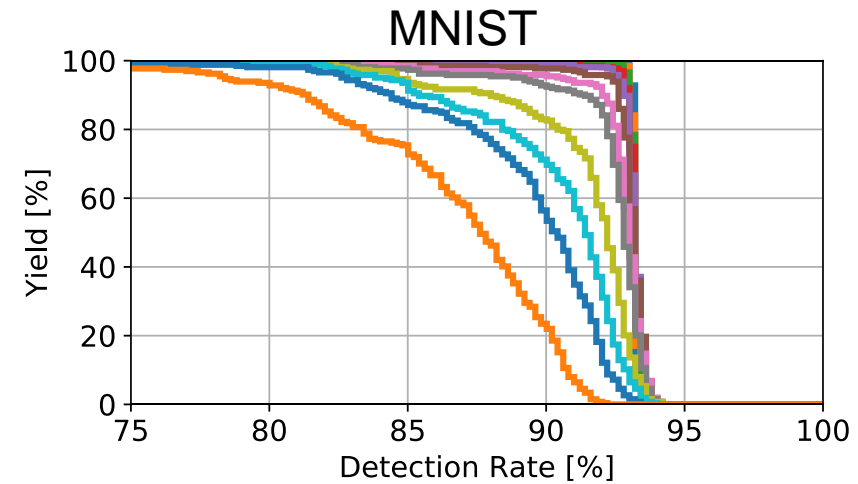
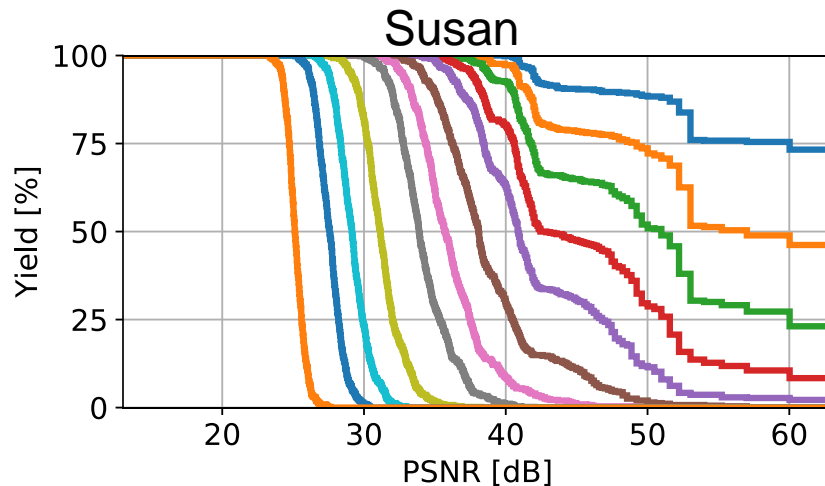
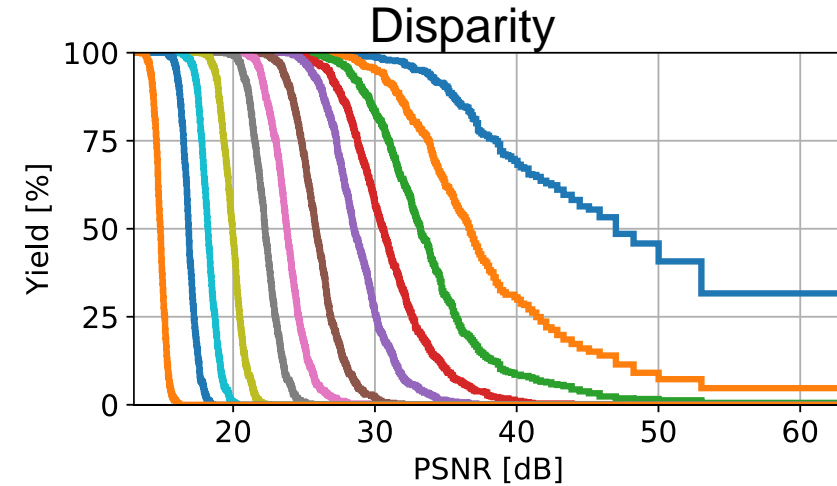
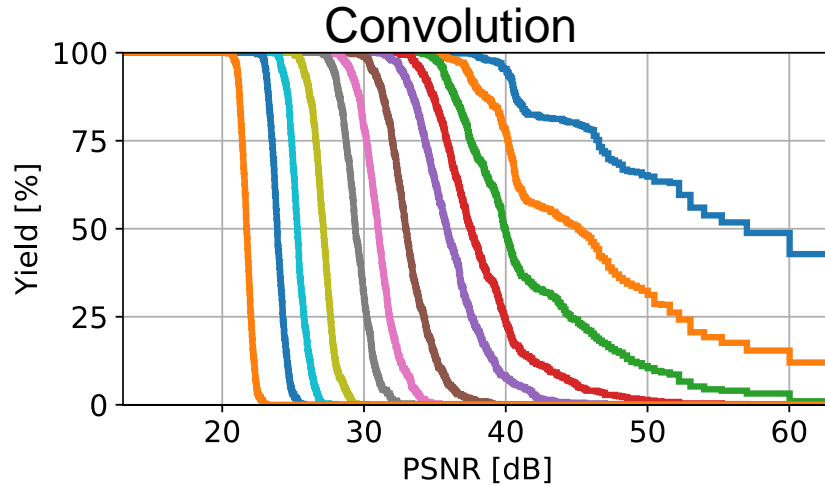


- Convolution benchmark: 4x refresh relaxation provides **~50% less memory accesses** with an **acceptable PSNR** of 35.1 dB

Results: Error-Resilience Yield Analysis

- Every benchmark is executed on 250 **randomly generated dies**
- The **yield** is plotted for different quality requirements and refresh rates

■ x 1.00 ■ x 1.50 ■ x 2.00 ■ x 2.50 ■ x 3.00 ■ x 4.00 ■ x 5.00 ■ x 6.00 ■ x 8.00 ■ x 10.00 ■ x 12.00 ■ x 16.00

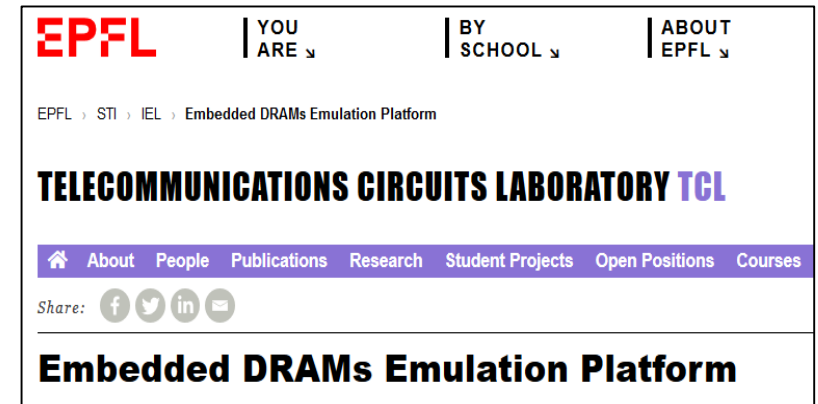


Conclusion

- **FPGA-based platform** for fast cycle-accurate emulation of unreliable memories
- Integrated into the **PULP platform**
- Accurate emulation of eDRAM under **sub-critical refresh** using **programmable data retention time maps**
- Evaluation of the **application output quality** and **application error resilience** through **yield analysis**
- Results show that refresh relaxation significantly reduces the number of access cycles with limited impact on quality

Platform available as **Xilinx Vivado IP core**: <https://tcl.epfl.ch/edramemu>

(PULP-based Platform available soon)



Appendix: eDRAM Emulator Architecture on FPGA

- **Goal:** FPGA model for **cycle-accurate emulation** of eDRAM memories under sub-critical refresh rates.
- **Approach:** **Programmable DRT map** to reflect process variations. Faults are modeled as **bit flips**.
- **Problem:** Logic for every bit cell does not scale.
- **Solution:** Keep track of last write, inject errors in read port.
- **Implementation:** SRAM blocks for data, timestamps and DRT map.

