Debugging Intermittent Systems

Brief overview of the current debugger landscape

January 24, 2024

Kevin Kollenda

Friedrich-Alexander-Universität Erlangen-Nürnberg

Chair in Distributed Systems
and Operating Systems

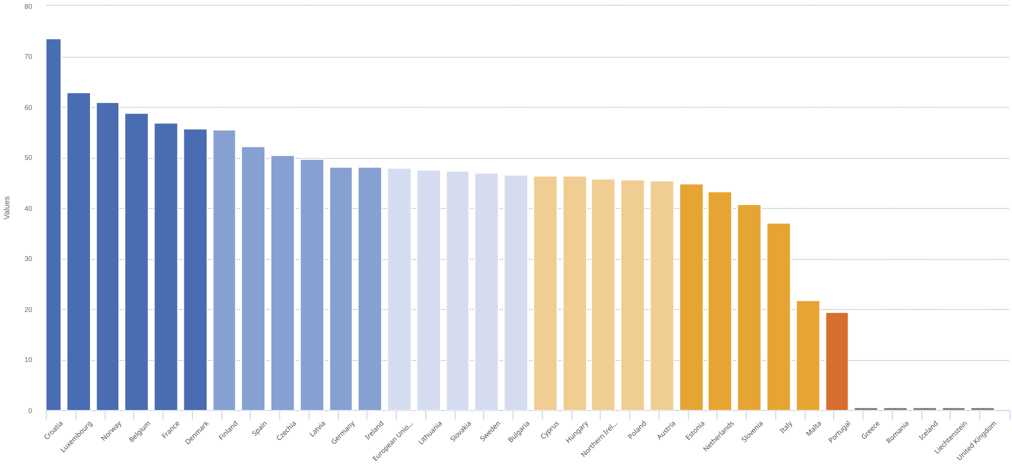
Friedrich-Alexander-Universität
Technische Fakultät

Motivation

Sales and collection of portable batteries and accumulators

Geopolitical entity (reporting) / Waste management operations Time frequency:**Annual** Waste categories:**Portable batteries and accumulators** Unit of measure:**Percentage** Time:**2021**.

Values forWaste collected. Bars in red represent not available data..



Sales and collection of portable batteries and accumulators

Source of data: Eurostat (online data code: env_waspb)
Last update 03/01/2024 23:00

eurostat

Table of Contents

Intermittent Systems

Intermittent Systems

- Removal of batteries in favor of (super)-capacitors

Intermittent Systems

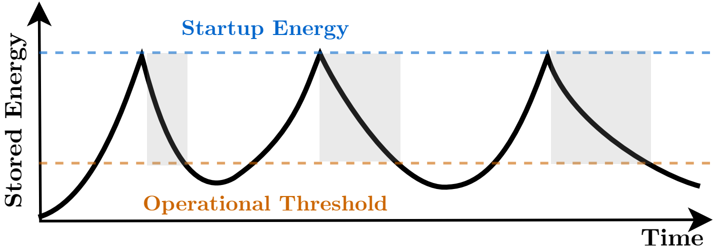
- Removal of batteries in favor of (super)-capacitors
- Harvest energy from external sources (solar, thermal, ...)

Intermittent Systems

- Removal of batteries in favor of (super)-capacitors
- Harvest energy from external sources (solar, thermal, ...)
- Execution is frequently interrupted due to power loss

Intermittent Systems

- Removal of batteries in favor of (super)-capacitors
- Harvest energy from external sources (solar, thermal, ...)
- Execution is frequently interrupted due to power loss



Intermittent Systems

How can we ensure reliable program execution considering the rapidly changing energy inputs?

Reliable Execution

- Checkpointing
  - Save volatile state to non-volatile memory
  - Restore from checkpoint after power loss

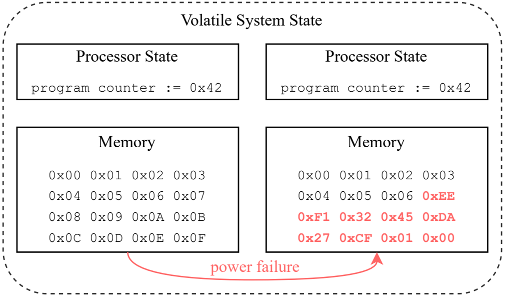
Reliable Execution

- Checkpointing
  - Save volatile state to non-volatile memory
  - Restore from checkpoint after power loss
- Task-Based Programming
  - Program is divided into *tasks*
  - Tasks are only run when there is enough energy available

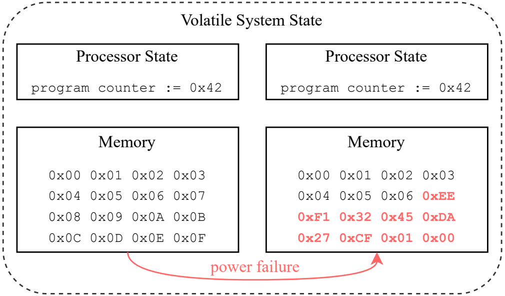
Reliable Execution

- Checkpointing
  - Save volatile state to non-volatile memory
  - Restore from checkpoint after power loss
- Task-Based Programming
  - Program is divided into *tasks*
  - Tasks are only run when there is enough energy available
- Non-Volatile Systems
  - Conventional DRAM can be replaced by NVRAM
  - Non-volatile microarchitectures for processors

Volatile State Restoration



Volatile State Restoration



Peripheral State Restoration

```
sensor = InitializeSensor();
Calibrate(sensor);
while(data = Read(sensor)) {
  Checkpoint();
  // <Power failure occurs>
  Transmit(data);
}
```

Debugging Challenges

Energy Behaviour

Common debugging methods *increase* the system's power draw:

Energy Behaviour

Common debugging methods *increase* the system's power draw:

- Toggling an LED upon reaching a certain line of code

Energy Behaviour

Common debugging methods *increase* the system's power draw:

- Toggling an LED upon reaching a certain line of code
- Tracing program flow using `printf`

Energy Behaviour

Common debugging methods *increase* the system's power draw:

- Toggling an LED upon reaching a certain line of code
- Tracing program flow using `printf`
- Streaming log output to external devices (serial, I²C, ...)

Energy Behaviour

Common debugging methods *increase* the system's power draw:

- Toggling an LED upon reaching a certain line of code
- Tracing program flow using `printf`
- Streaming log output to external devices (serial, I^2C, ...)
- Assertions for simple invariants and complex data structures

Energy Behaviour

Common debugging methods *increase* the system's power draw:

- Toggling an LED upon reaching a certain line of code
- Tracing program flow using `printf`
- Streaming log output to external devices (serial, I^2C, ...)
- Assertions for simple invariants and complex data structures

Established embedded system debuggers do not account for this and require that the device under test is *continuously* powered.

Energy Behaviour

Common debugging methods *increase* the system's power draw:

- Toggling an LED upon reaching a certain line of code
- Tracing program flow using `printf`
- Streaming log output to external devices (serial, I²C, ...)
- Assertions for simple invariants and complex data structures

Established embedded system debuggers do not account for this and require that the device under test is *continuously* powered.

⇒ **Intermittent systems require purpose-built energy-aware debuggers**

Software-Based Debugger Issues

Snippet (a)

```
Checkpoint();
total = NVM_Load();
for i < N {


  total += Sense();
  NVM_Store(total);
}
// i gets saved
Checkpoint();
```

Software-Based Debugger Issues

Snippet (a)

```
Checkpoint();
total = NVM_Load();
for i < N {


  total += Sense();
  NVM_Store(total);
}
// i gets saved
Checkpoint();
```

Snippet (b)

```
Checkpoint();
total = NVM_Load();
for i < N {
  // i gets saved
  DBG_Breakpoint();
  total += Sense();
  NVM_Store(total);
}

Checkpoint();
```

Software-Based Debugger Issues

Snippet (a)

```
Checkpoint();
total = NVM_Load();
for i < N {


  total += Sense();
  NVM_Store(total);
}
// i gets saved
Checkpoint();
```

Snippet (b)

```
Checkpoint();
total = NVM_Load();
for i < N {
  // i gets saved
  DBG_Breakpoint();
  total += Sense();
  NVM_Store(total);
}

Checkpoint();
```

Software-based debuggers can alter the program's behaviour!

Requirements for an Intermittent System Debugger

Intermittent system debuggers must not only provide *energy-neutrality* for existing debugging operations, like

Requirements for an Intermittent System Debugger

Intermittent system debuggers must not only provide *energy-neutrality* for existing debugging operations, like

- Breakpoints with single-stepping

Requirements for an Intermittent System Debugger

Intermittent system debuggers must not only provide *energy-neutrality* for existing debugging operations, like

- Breakpoints with single-stepping
- Output tracing & assertions

Requirements for an Intermittent System Debugger

Intermittent system debuggers must not only provide *energy-neutrality* for existing debugging operations, like

- Breakpoints with single-stepping
- Output tracing & assertions
- Reading from and writing to device memory

Requirements for an Intermittent System Debugger

Intermittent system debuggers must not only provide *energy-neutrality* for existing debugging operations, like

- Breakpoints with single-stepping
- Output tracing & assertions
- Reading from and writing to device memory

but also be able to *manipulate the device's energy input* to enable the

Requirements for an Intermittent System Debugger

Intermittent system debuggers must not only provide *energy-neutrality* for existing debugging operations, like

- Breakpoints with single-stepping
- Output tracing & assertions
- Reading from and writing to device memory

but also be able to *manipulate the device's energy input* to enable the

- Recording of the device's energy consumption

Requirements for an Intermittent System Debugger

Intermittent system debuggers must not only provide *energy-neutrality* for existing debugging operations, like

- Breakpoints with single-stepping
- Output tracing & assertions
- Reading from and writing to device memory

but also be able to *manipulate the device's energy input* to enable the

- Recording of the device's energy consumption
- Manual injection of power failures

Requirements for an Intermittent System Debugger

Intermittent system debuggers must not only provide *energy-neutrality* for existing debugging operations, like

- Breakpoints with single-stepping
- Output tracing & assertions
- Reading from and writing to device memory

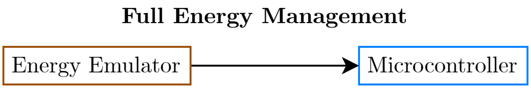
but also be able to *manipulate the device's energy input* to enable the

- Recording of the device's energy consumption
- Manual injection of power failures
- Replay of previously captured energy traces

Intermittent Systems Debugging

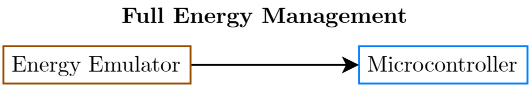
Energy Emulation Comparison

Choice #1



Full energy emulation:

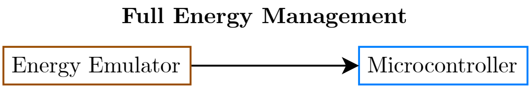
Energy Emulation Comparison



Full energy emulation:

- Replaces device's power supply

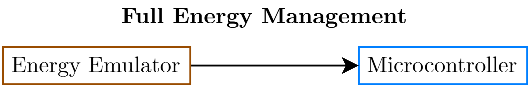
Energy Emulation Comparison



Full energy emulation:

- Replaces device's power supply
- Enables energy trace replay

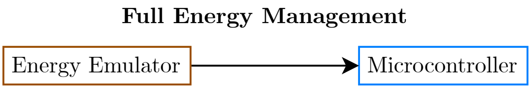
Energy Emulation Comparison

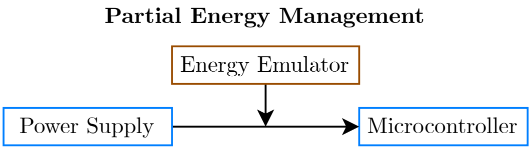


Full energy emulation:

- Replaces device's power supply
- Enables energy trace replay
- Simulated components

Energy Emulation Comparison



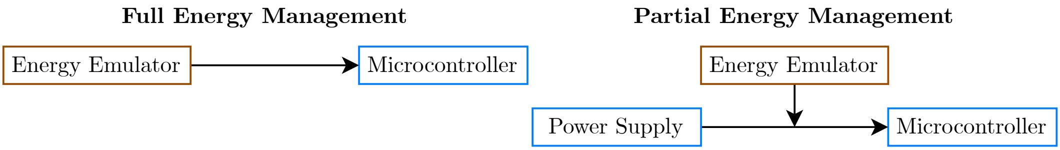


Full energy emulation:

- Replaces device's power supply
- Enables energy trace replay
- Simulated components

Partial energy emulation:

Energy Emulation Comparison



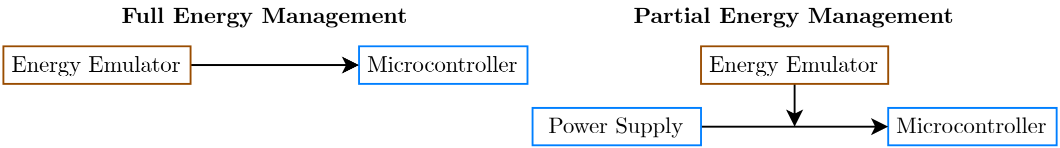
Full energy emulation:

- Replaces device's power supply
- Enables energy trace replay
- Simulated components

Partial energy emulation:

- Hooks into existing circuitry

Energy Emulation Comparison



Full energy emulation:

- Replaces device's power supply
- Enables energy trace replay
- Simulated components

Partial energy emulation:

- Hooks into existing circuitry
- Closer to real-world conditions

Debugger Design

Software-based debuggers:

Debugger Design

Software-based debuggers:

- Offer software library

Software-based debuggers:

- Offer software library
- Impact program behaviour

Software-based debuggers:

- Offer software library
- Impact program behaviour
- Low barrier of entry

Debugger Design

Software-based debuggers:

- Offer software library
- Impact program behaviour
- Low barrier of entry

Hardware-based debuggers:

Debugger Design

Software-based debuggers:

- Offer software library
- Impact program behaviour
- Low barrier of entry

Hardware-based debuggers:

- Connect to processor's in-built debugging circuitry

Debugger Design

Software-based debuggers:

- Offer software library
- Impact program behaviour
- Low barrier of entry

Hardware-based debuggers:

- Connect to processor’s in-built debugging circuitry
- Debugging tasks are offloaded

Debugger Design

Software-based debuggers:

- Offer software library
- Impact program behaviour
- Low barrier of entry

Hardware-based debuggers:

- Connect to processor's in-built debugging circuitry
- Debugging tasks are offloaded
- Increased energy consumption

Debugger Design

Software-based debuggers:

- Offer software library
- Impact program behaviour
- Low barrier of entry

Hardware-based debuggers:

- Connect to processor's in-built debugging circuitry
- Debugging tasks are offloaded
- Increased energy consumption

Regardless of the debugger's kind:

- Standalone or built upon existing debuggers (i.e. GDB)
- Energy management interface

Energy-Neutrality

Intrusive debugging always consumes additional energy.

Energy-Neutrality

Intrusive debugging always consumes additional energy.

How can we achieve realistic conditions during debugging?

Energy-Neutrality

Intrusive debugging always consumes additional energy.

How can we achieve realistic conditions during debugging?

Energy-Guards [1]

Neutralize the energy impact of certain actions or code snippets.

Energy-Neutrality

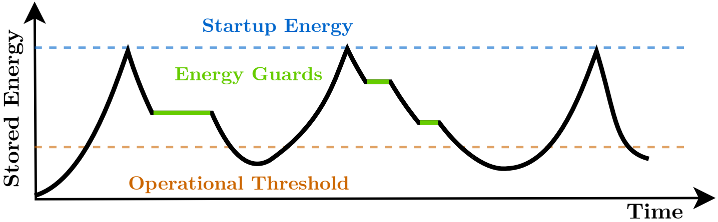
Intrusive debugging always consumes additional energy.

How can we achieve realistic conditions during debugging?

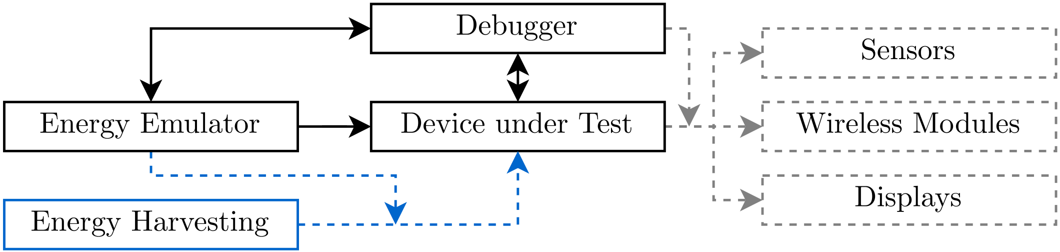
Energy-Guards [1]

Neutralize the energy impact of certain actions or code snippets.

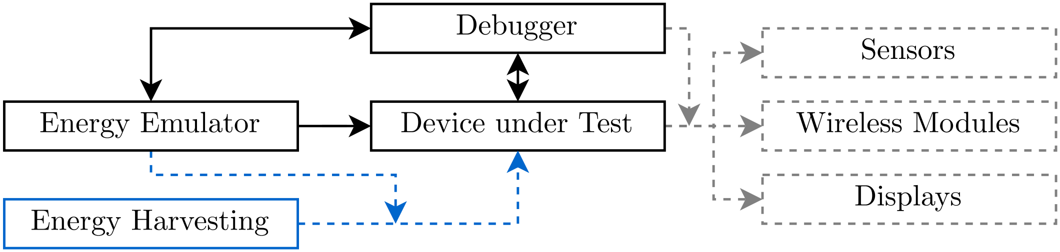
In practice:



Bringing it all together

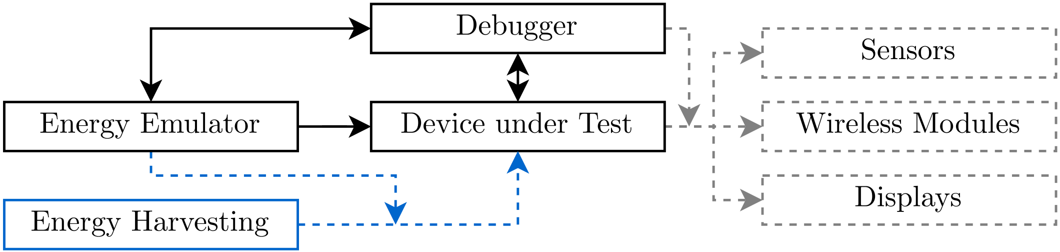


Bringing it all together



→ Mask energy footprint of complex assertions

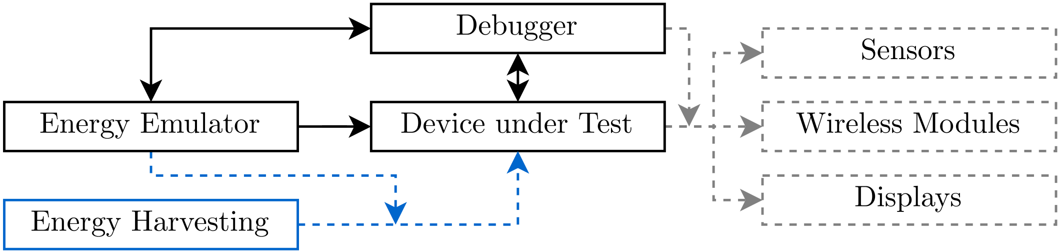
Bringing it all together



→ Mask energy footprint of complex assertions

→ Pause energy consumption during breakpoints

Bringing it all together

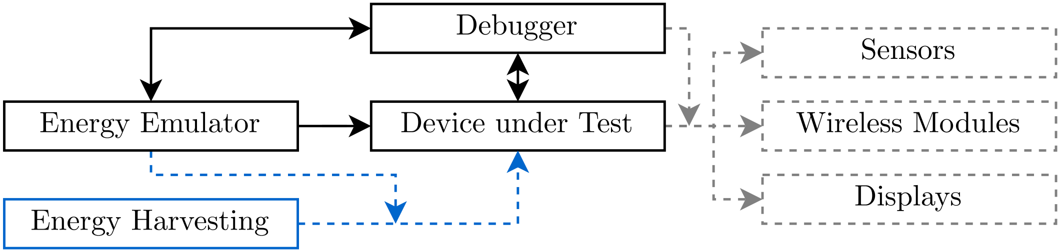


→ Mask energy footprint of complex assertions

→ Pause energy consumption during breakpoints

→ Recreate previously recorded energy environments

Bringing it all together



→ Mask energy footprint of complex assertions

→ Pause energy consumption during breakpoints

→ Recreate previously recorded energy environments

⇒ **Debug intermittent systems like regular embedded systems**

Existing Solutions & Further Research

Energy-Interference-Free Debugger (EDB) 2016

Energy-Interference-Free Debugger (EDB) 2016

- Hooks into existing energy circuit

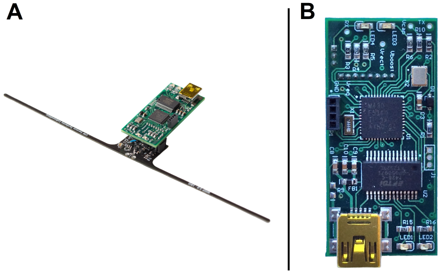
Energy-Interference-Free Debugger (EDB) 2016

- Hooks into existing energy circuit
- Provides software library for debugging

Energy-Interference-Free Debugger (EDB)

2016

- Hooks into existing energy circuit
- Provides software library for debugging
- First available intermittent system debugger



[1]

Debugger for Intermittently-Powered Systems (DIPS) 2022

Debugger for Intermittently-Powered Systems (DIPS) 2022

- Fully manipulates the device's energy input

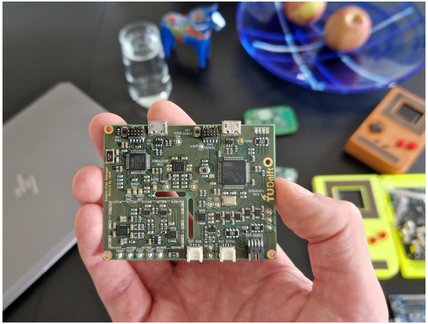
Existing Debuggers

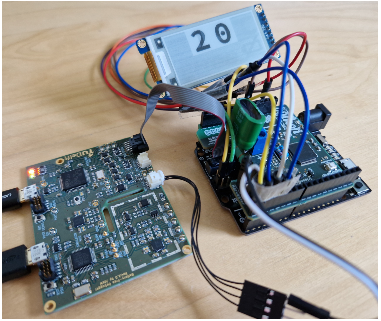
Debugger for Intermittently-Powered Systems (DIPS) 2022

- Fully manipulates the device’s energy input
- Utilizes in-built debugging circuitry

Debugger for Intermittently-Powered Systems (DIPS) 2022

- Fully manipulates the device's energy input
- Utilizes in-built debugging circuitry
- Scriptable interface for automatic testing





[2]

Existing Debuggers

	EDB	DIPS
Debugger Design		
Energy Management		
GDB-Based		
Energy-neutral Debugging		
Breakpoints		
Automated Testing		
Single Stepping		
Supported Architectures		

Existing Debuggers

	EDB	DIPS
Debugger Design	Software	
Energy Management	Partial	
GDB-Based	No	
Energy-neutral Debugging	Yes	
Breakpoints	Software	
Automated Testing	No	
Single Stepping	No	
Supported Architectures	MSP430	

Existing Debuggers

	EDB	DIPS
Debugger Design	Software	Hardware
Energy Management	Partial	Full
GDB-Based	No	Yes
Energy-neutral Debugging	Yes	Yes
Breakpoints	Software	Software & Hardware
Automated Testing	No	Yes
Single Stepping	No	Yes
Supported Architectures	MSP430	ARM

Future Research

- Support additional architectures

Future Research

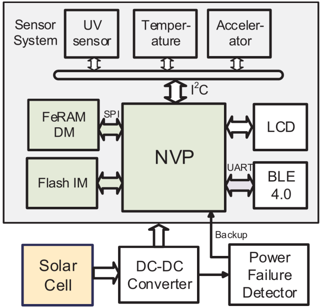
- Support additional architectures
- Improve energy emulation hardware

Future Research

- Support additional architectures
- Improve energy emulation hardware
- Incorporate existing testing frameworks (i.e. fuzzing, ...)

Future Research

- Support additional architectures
- Improve energy emulation hardware
- Incorporate existing testing frameworks (i.e. fuzzing, ...)
- Progress in non-volatile technologies lessen impact of intermittency



[3]

Conclusion

Summary

- Intermittent systems pose unique challenges to existing debuggers

Summary

- Intermittent systems pose unique challenges to existing debuggers
  - Energy-neutral debugging via energy-guards

Summary

- Intermittent systems pose unique challenges to existing debuggers
  - Energy-neutral debugging via energy-guards
  - Real-world energy conditions provided by energy emulator

Summary

- Intermittent systems pose unique challenges to existing debuggers
  - Energy-neutral debugging via energy-guards
  - Real-world energy conditions provided by energy emulator
- Requires tight integration between energy emulator and debugger

Summary

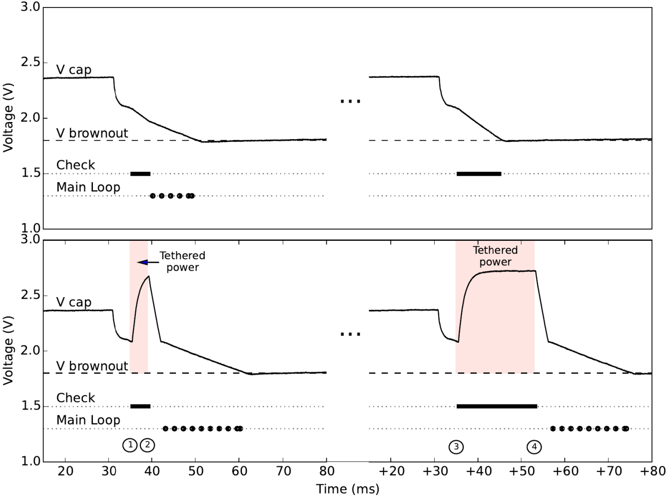
- Intermittent systems pose unique challenges to existing debuggers
  - Energy-neutral debugging via energy-guards
  - Real-world energy conditions provided by energy emulator
- Requires tight integration between energy emulator and debugger
- Bright future for intermittent devices

Summary

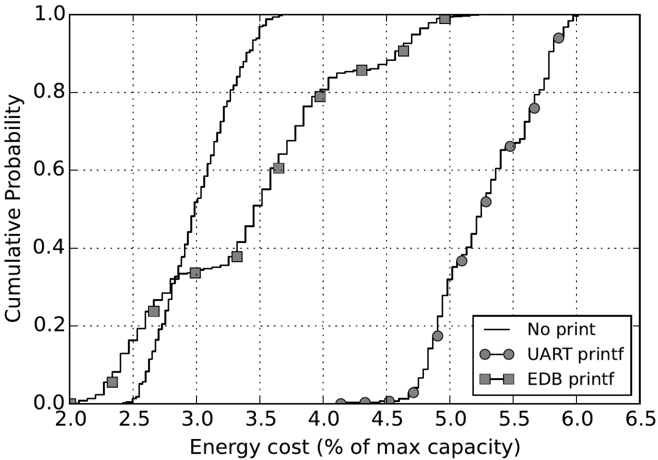
- Intermittent systems pose unique challenges to existing debuggers
  - Energy-neutral debugging via energy-guards
  - Real-world energy conditions provided by energy emulator
- Requires tight integration between energy emulator and debugger
- Bright future for intermittent devices

$\Rightarrow$ Increase IoT sustainability by reducing the need for batteries

Questions?



EDB providing assertions with power using energy-guards [1]



Impact of guarded `printf` calls [1]

Device Under Test	t_{init} (ms)	t_{rec} (ms)
nRF52 [Arm-M4] [35]	311.1	72.7
SAM4L8 [Arm-M4] [32]	324.7	75.8
MKL05Z [Arm-M0+] [36]	309.6	105.8
STM32F3 [Arm M4] [49]	318.6	68.2
Apollo 3 [Arm M4] [47]	331.1	95.6

DIPS initial and reconnection latencies [2]

References (1)

A. Colin, G. Harvey, B. Lucia, and A. P. Sample.
An energy-interference-free hardware-software debugger for intermittent energy-harvesting systems.
In *Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '16, page 577–589, New York, NY, USA, 2016. Association for Computing Machinery.

J. de Winkel, T. Hoefnagel, B. Blokland, and P. Pawełczak.
Dips: Debug intermittently-powered systems like any embedded system.

References (2)

In *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems*, SenSys '22, page 222–235, New York, NY, USA, 2023. Association for Computing Machinery.

Y. Liu, Z. Li, H. Li, Y. Wang, X. Li, K. Ma, S. Li, M.-F. Chang, S. John, Y. Xie, J. Shu, and H. Yang.
Ambient energy harvesting nonvolatile processors: From circuit to system.
In *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, 2015.