An Energy-interference-free Hardware-Software Debugger for Intermittent Energy-harvesting Systems

Alexei Colin*^ Graham Harvey*
Alanson Sample* Brandon Lucia^

*Disney Research Pittsburgh
^Dept. of ECE, Carnegie Mellon University

Carnegie Mellon University
Electrical & Computer ENGINEERING

Disney Research, Pittsburgh

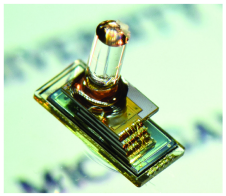
Outline

- Energy-harvesting devices
- Debugging intermittent programs
- Energy-interference-free Debugger (EDB)
- Evaluation: debugging with EDB
- Related work and conclusion

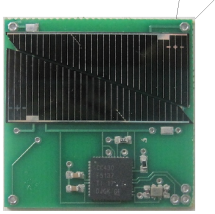
Applications of Energy-Harvesting

- Power devices with energy from the environment
- Eliminate batteries and wires for tiny form factors
- Embed computation into spaces, things, or bodies

Thermal	:	*in-body sensor*
Mechanical	:	*infrastructure monitor*
Radio waves	:	*inventory tracking*
Solar	*:*	*nano-satellite*



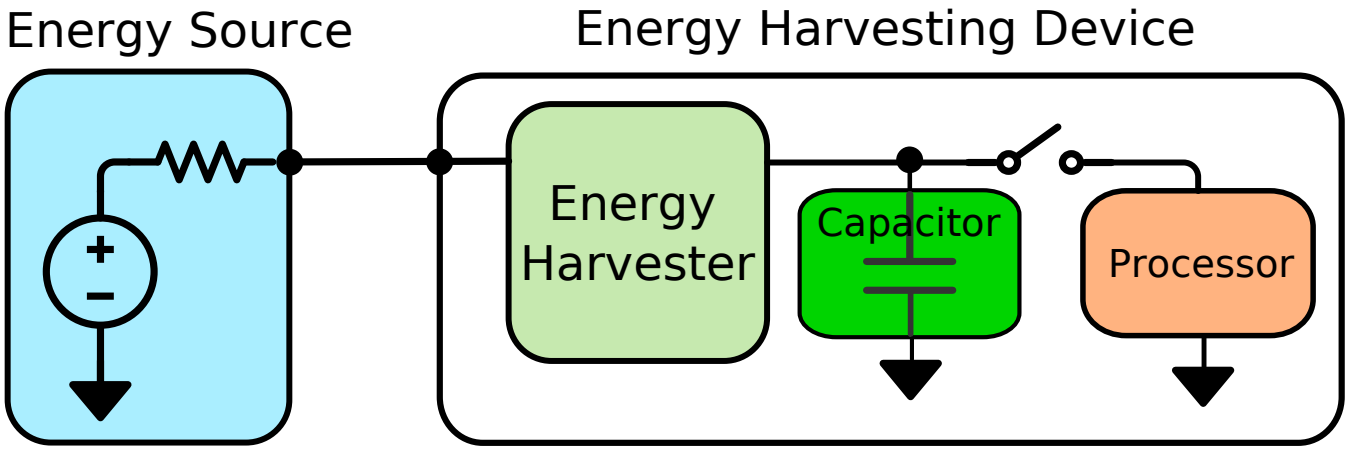
M3 Mote



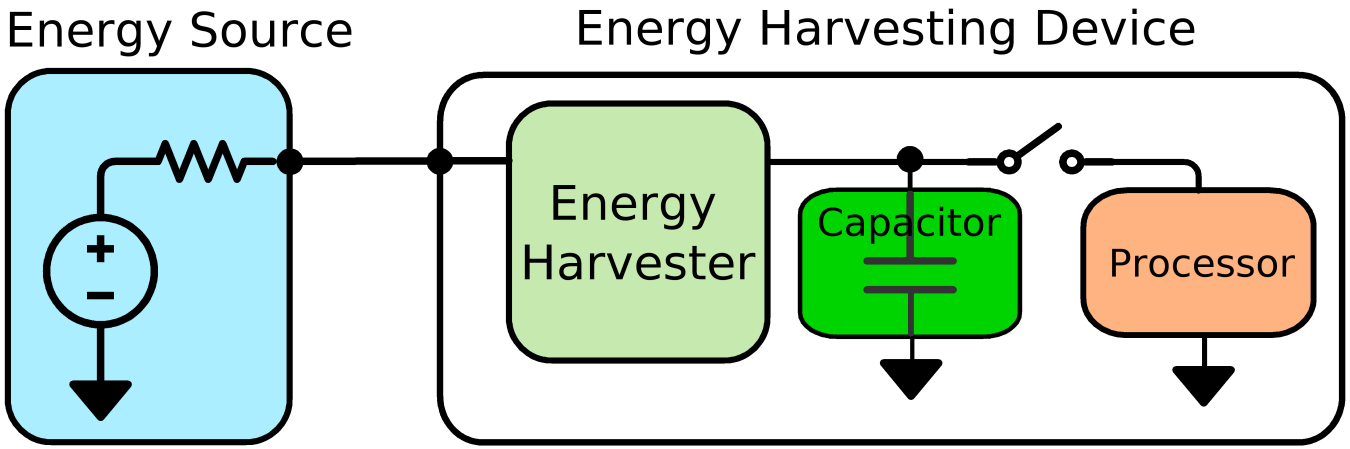
KickSat

Application = energy source + hardware + **software**

Anatomy of an Energy-Harvesting Computer



Anatomy of an Energy-Harvesting Computer



WISP: Wireless Identification and Sensing Platform*

Antenna

Harvester

Capacitor

Radio waves

Micro-controller

Volatile Memory

Non-volatile Memory

Thermo-meter

Accelerometer

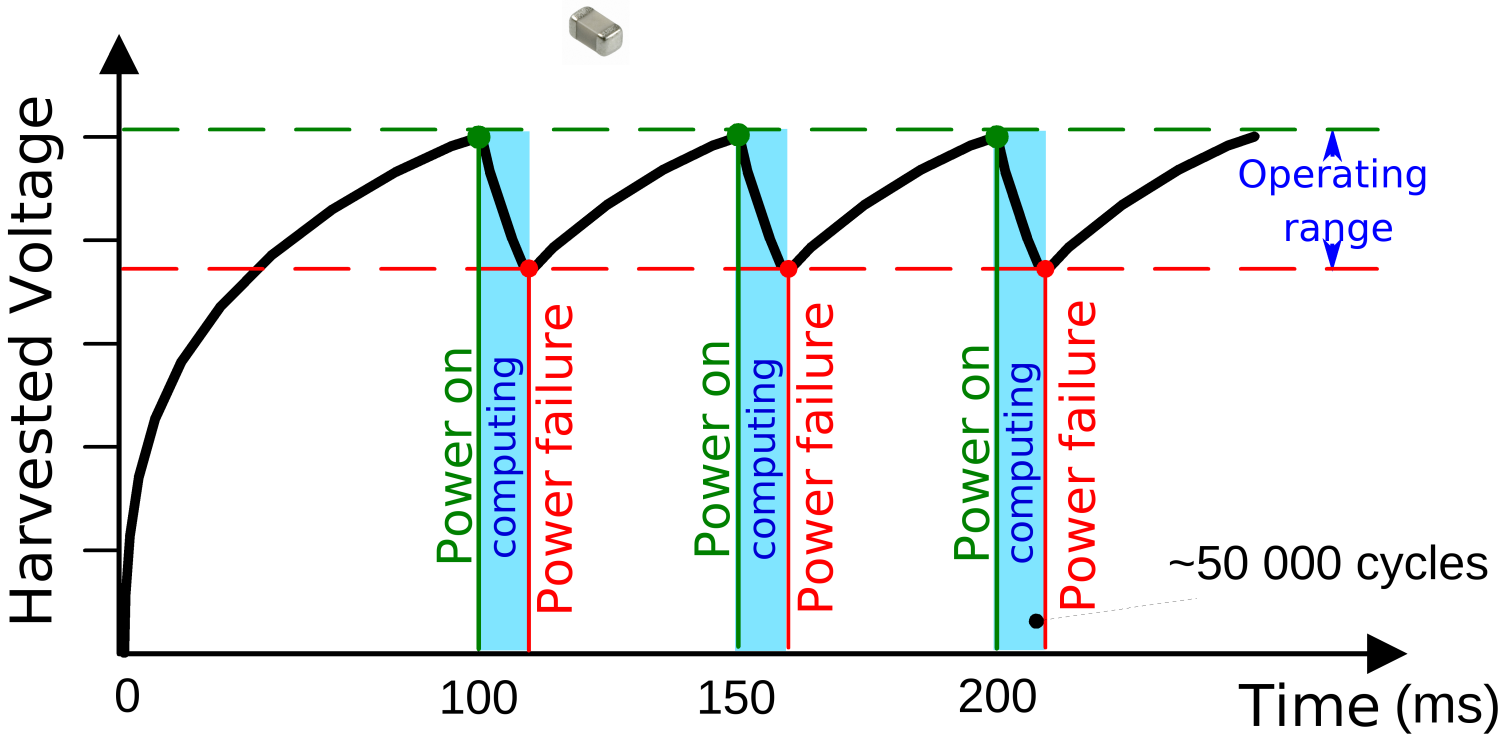
* A. Sample et. al. IEEE TIM, 2008

Intermittent Execution

Energy is available **intermittently**

Program runs **intermittently**

- Energy source is unreliable
- Energy storage is small (~ 2 mm^3)
- Execution may stop at any point
- All volatile state is lost on reboot

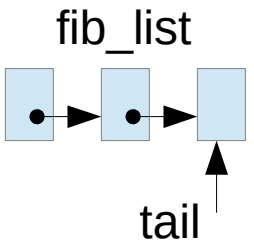


A Bug in an Intermittent Program

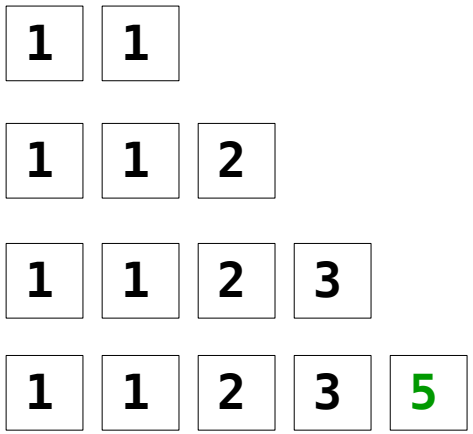
Source Code: fibonacci.c

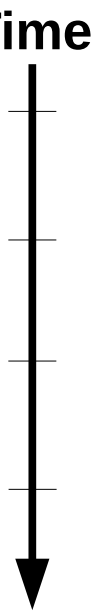
```
main {
  while (TRUE) {
    m, n = get_last_two_elements(fib_list)
    append_element(fib_list, m + n)
  }
}
```

Non-volatile memory



Expected program state



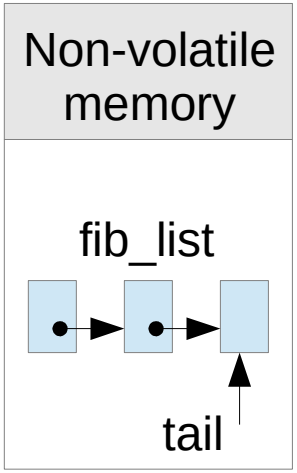


Observed program state

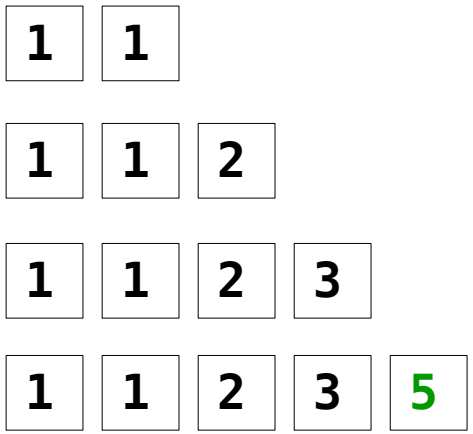
A Bug in an Intermittent Program

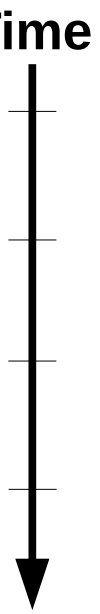
Source Code: fibonacci.c

```
main {
  while (TRUE) {
    m, n = get_last_two_elements(fib_list)
    append_element(fib_list, m + n)
  }
}
```

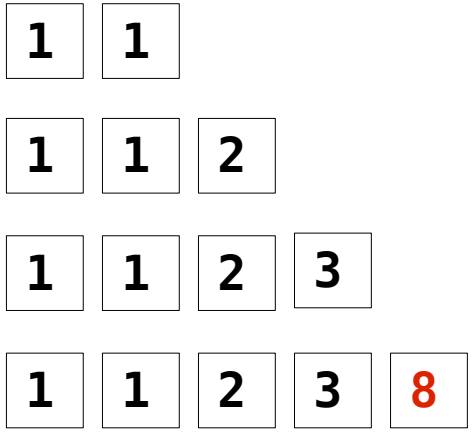


Expected program state





Observed program state



A Bug in an Intermittent Program

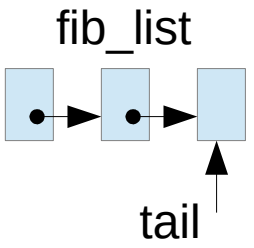
Source Code: fibonacci.c

```
main {
  while (TRUE) {
    m, n = get_last_two_elements(fib_list)
    append_element(fib_list, m + n)
  }
}
```

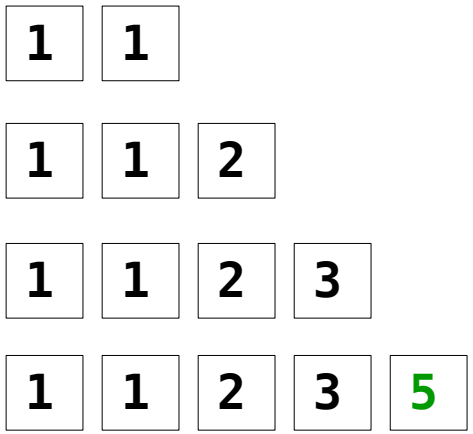
reboot

implementation is not correct if interrupted

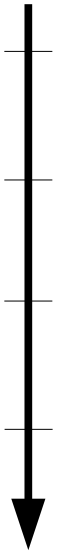
Non-volatile memory



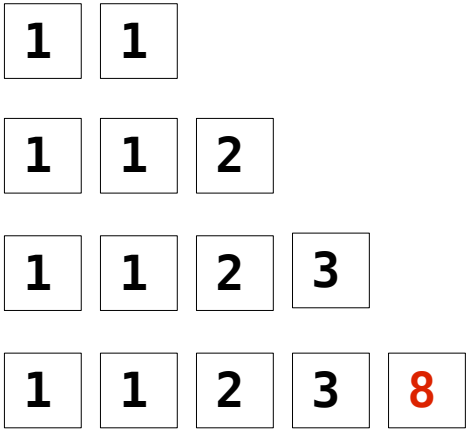
Expected program state



Time



Observed program state



Debugging an Intermittent Program

Option A

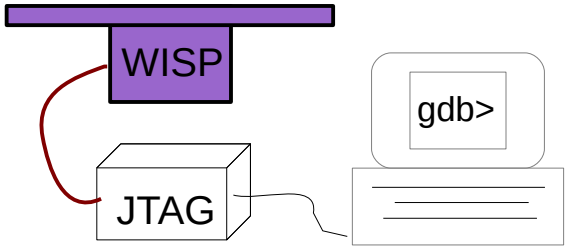
Option B

Option C

Debugging an Intermittent Program

Option A

JTAG debugger



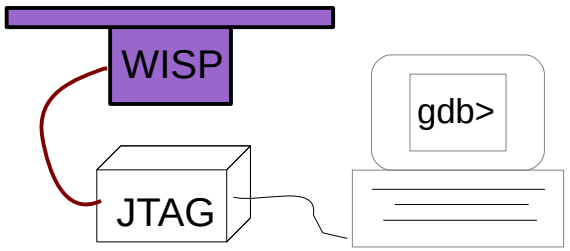
Hides bugs

Option B

Option C

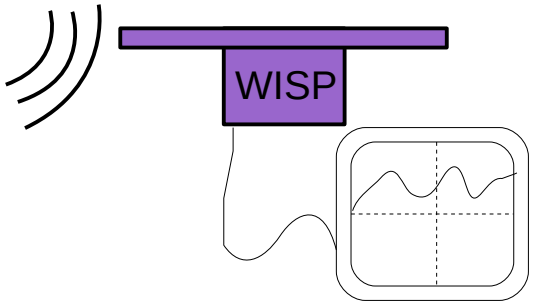
Debugging an Intermittent Program

Option A
JTAG debugger



Hides bugs

Option B
Oscilloscope

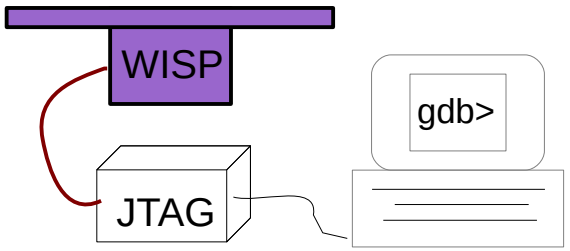


No program state

Option C

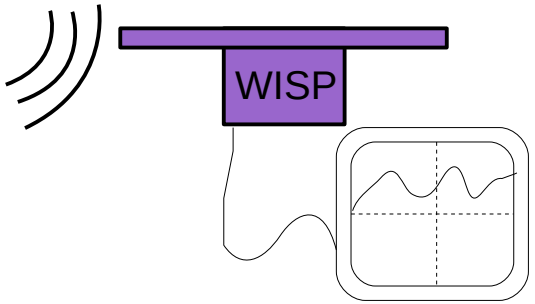
Debugging an Intermittent Program

Option A
JTAG debugger



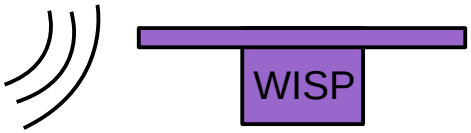
Hides bugs

Option B
Oscilloscope



No program state

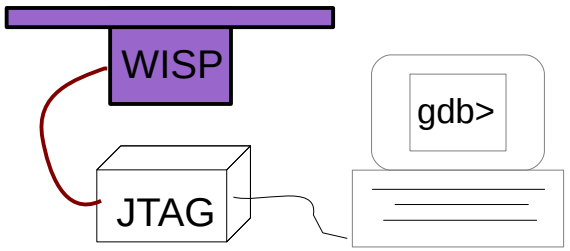
Option C
log events on device



Costs energy and space

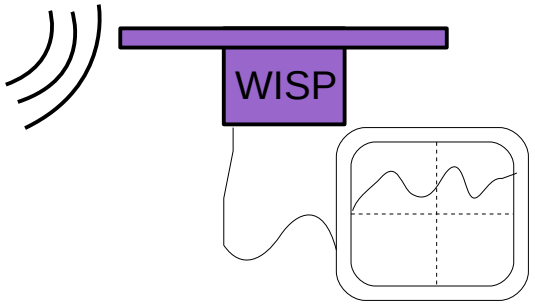
Debugging an Intermittent Program

Option A
JTAG debugger



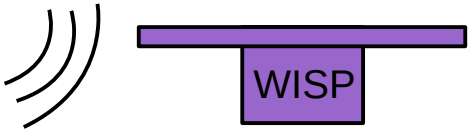
Hides bugs

Option B
Oscilloscope



No program state

Option C
log events on device



Costs energy and space

Interactive

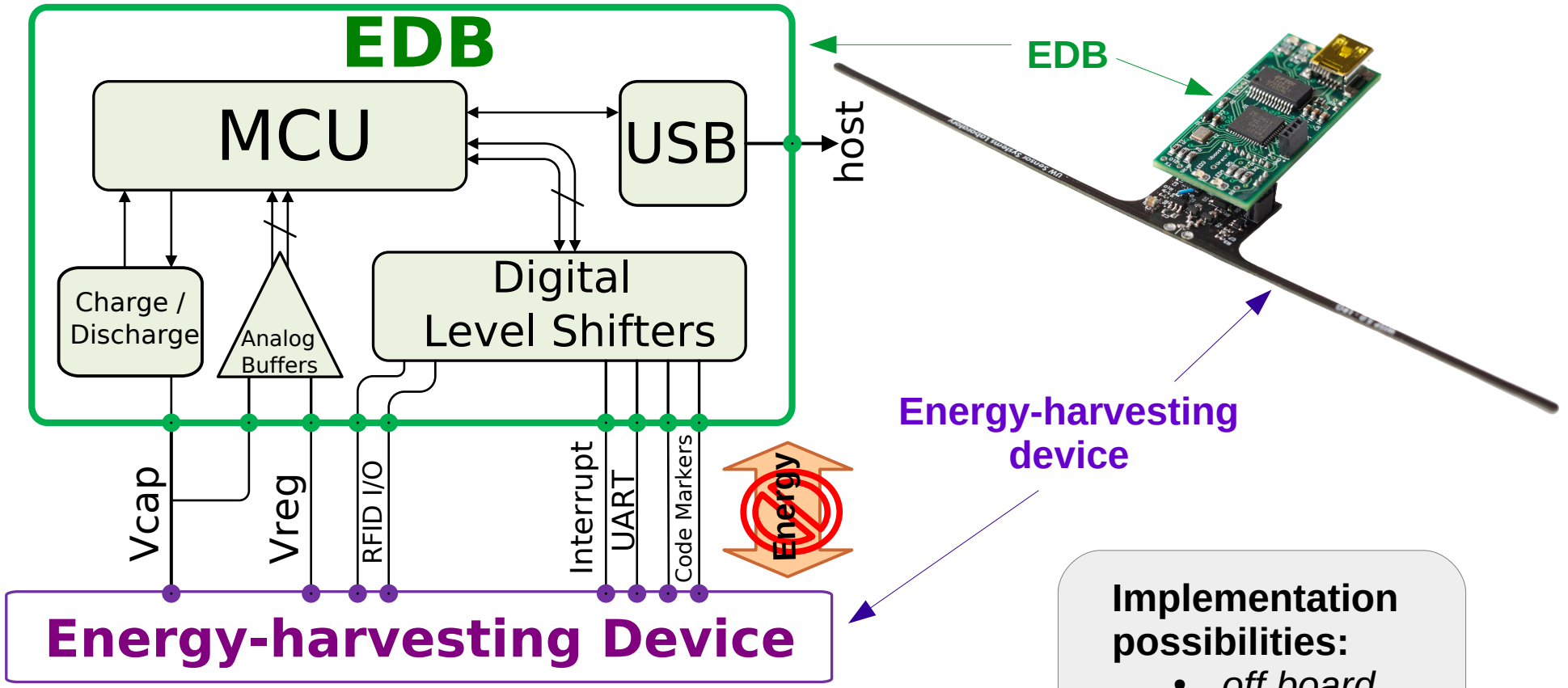
Inter-ference-free

Passive

EDB

Our main contribution is the *design and implementation of EDB*, an energy-interference-free debugger for intermittent energy-harvesting platforms.

EDB: Energy-interference-free Debugger



Implementation possibilities:

- *off board*
- on board
- in silicon

EDB Capabilities

Properties

EDB is energy-interference-free

Mechanisms

Passive

Trace energy level

Trace program events

Decode I/O messages

Active

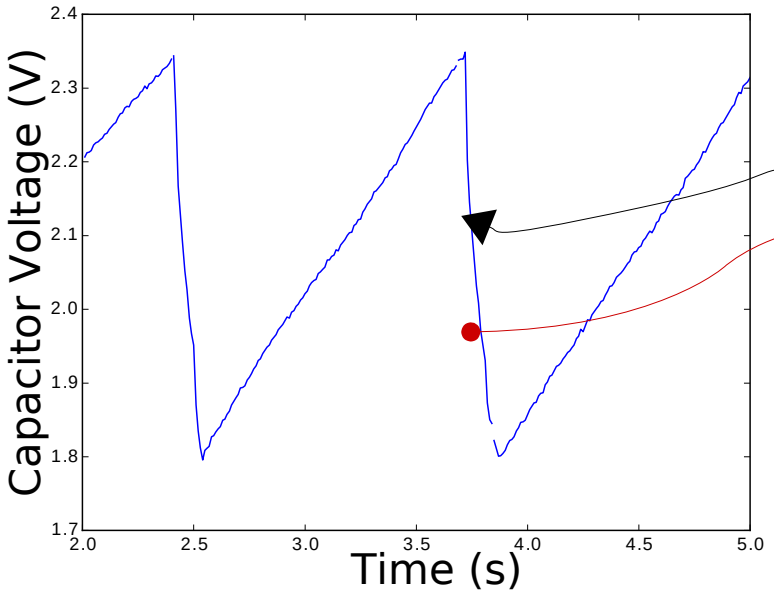
Save energy level

Set energy level

Supply power

EDB Joins Energy and Program Events

Energy State ⟺ Program State



EDB console

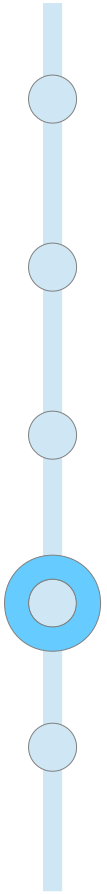
```
> break at line 42
> break at voltage 1.85V
> run
BREAKPOINT at line 42, V=2.05V

> continue
BREAKPOINT at line 64, V=1.85V
```

EDB enables new debugging primitives

Energy Guards	Keep-alive Assertions	Energy-aware Watchpoints	I/O Monitors

Outline

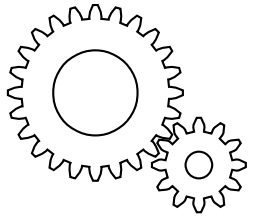
- Energy-harvesting devices
- Debugging intermittent programs
- Energy-interference-free Debugger (EDB)
- Evaluation: debugging with EDB
- Related work and conclusion



Evaluation Roadmap

- Case studies: applying EDB to debugging tasks

  (I) Zero-cost instrumentation
  (II) Keep-alive assertions
  (III) I/O monitoring



- Quantifying energy-interference

  - Leakage current
  - Save-restore resolution

Evaluation Case-Study (I): Zero-cost Instrumentation

broken_fibonacci.c

```
main:



  while True:
    append(list, value)
```

- **Program:** generate a list of Fibonacci values

Evaluation Case-Study (I): Zero-cost Instrumentation

broken_fibonacci.c

```
main:



  while True:
    append(list, value)
```

- **Program:** generate a list of Fibonacci values
- **Problem**: invalid values observed *on harvested energy*

1	1	2	3	8

Evaluation Case-Study (I): Zero-cost Instrumentation

broken_fibonacci.c

```
main:



  while True:
    append(list, value)
```

- **Program:** generate a list of Fibonacci values
- **Problem**: invalid values observed *on harvested energy*

1	1	2	3	8

- **Approach**: add an invariant check

$$n_i = n_{i-2} + n_{i-1}$$

Evaluation Case-Study (I): Zero-cost Instrumentation

broken_fibonacci.c

```
main:

 for value in list:
   assert(value is valid)

 while True:
   append(list, value)
```

} invariant check on reboot

- **Program:** generate a list of Fibonacci values
- **Problem**: invalid values observed *on harvested energy*

1	1	2	3	8

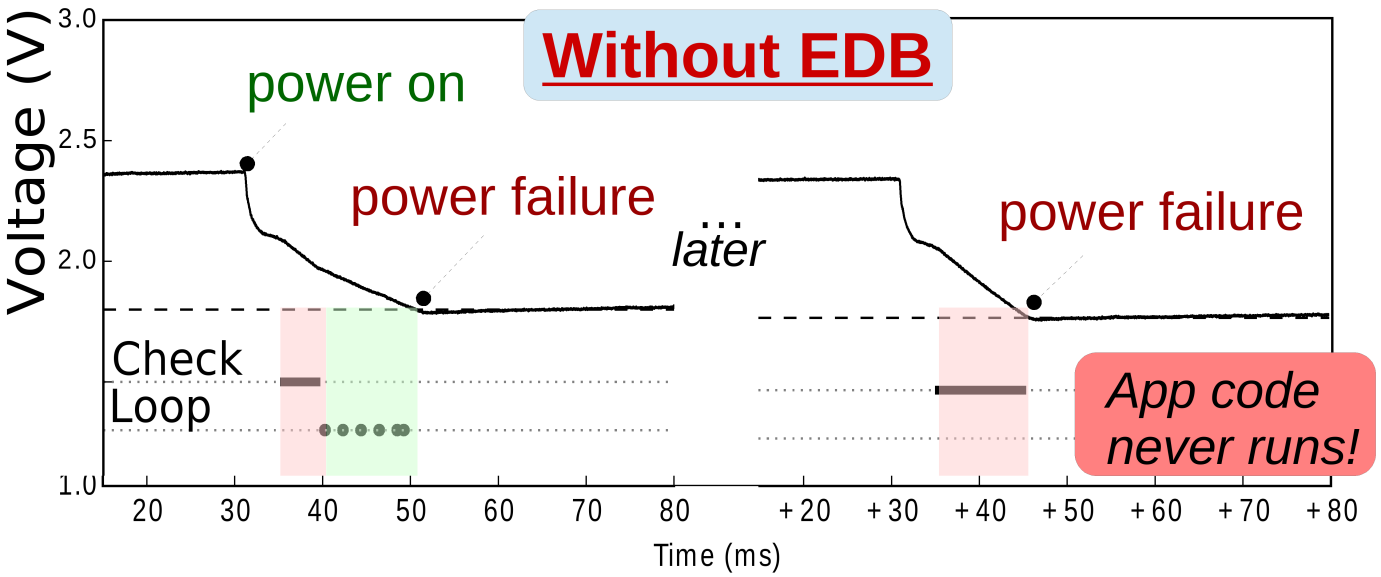
- **Approach**: add an invariant check

$$n_i = n_{i-2} + n_{i-1}$$

Evaluation Case-Study (I): Zero-cost Instrumentation (cont.)

broken_fibonacci.c

```
main:

  for value in list:
     assert(...)

  while True:
    append(...)
```

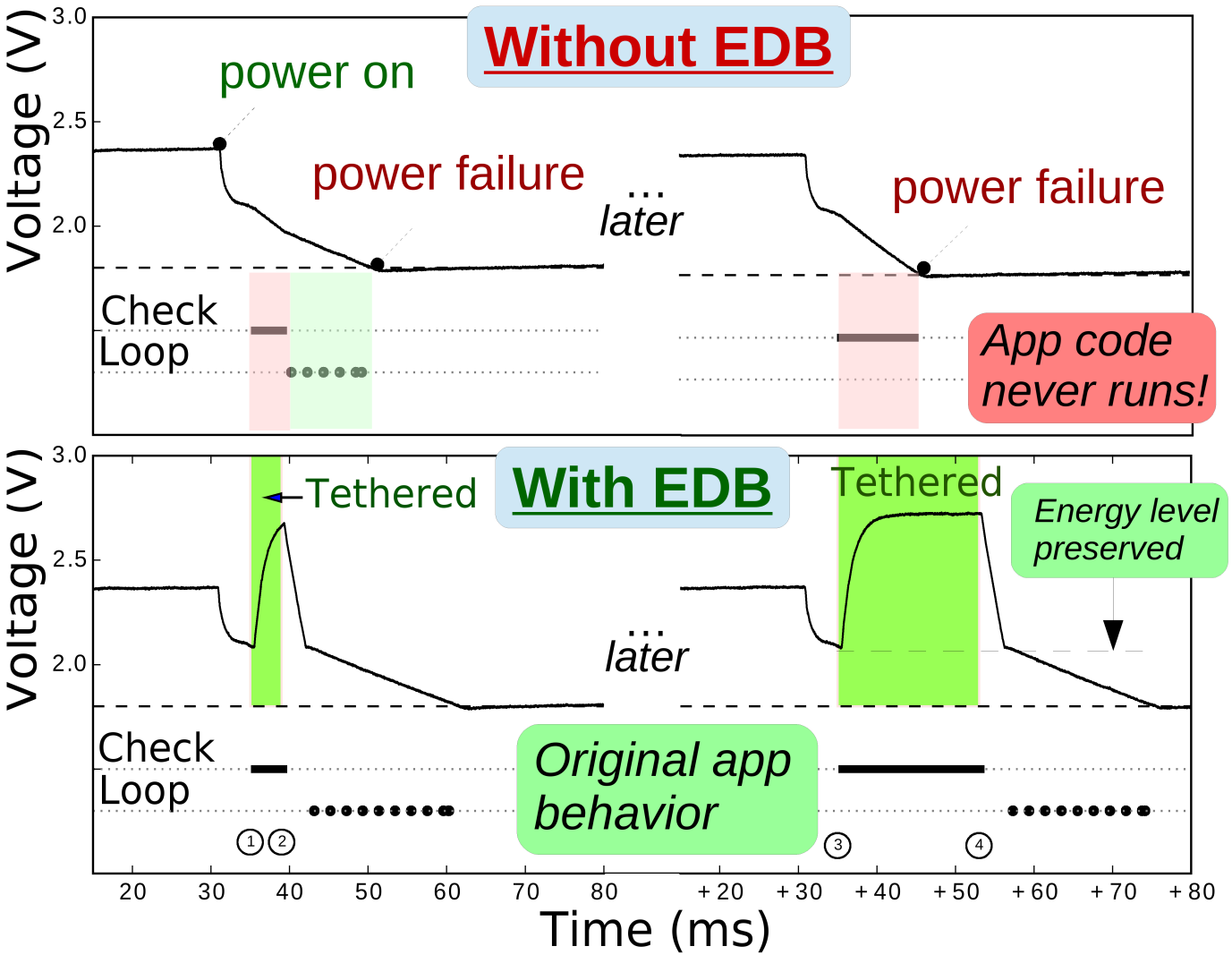


Evaluation Case-Study (I): Zero-cost Instrumentation (cont.)

broken_fibonacci.c

```
main:
  energy_guard {
  for value in list:
     assert(...)
  }
  while True:
    append(...)
```

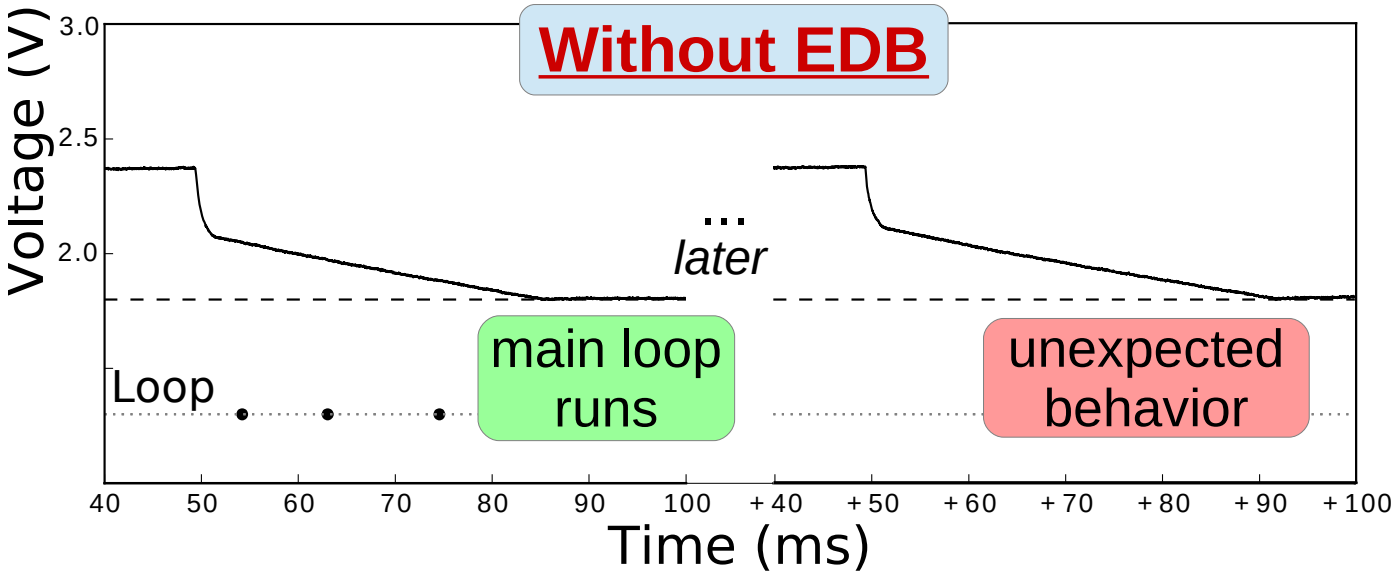
EDB *energy guards* hide the energy cost of instrumentation code.



Evaluation Case-Study (II): Keep-Alive Assertions

broken_fibonacci.c

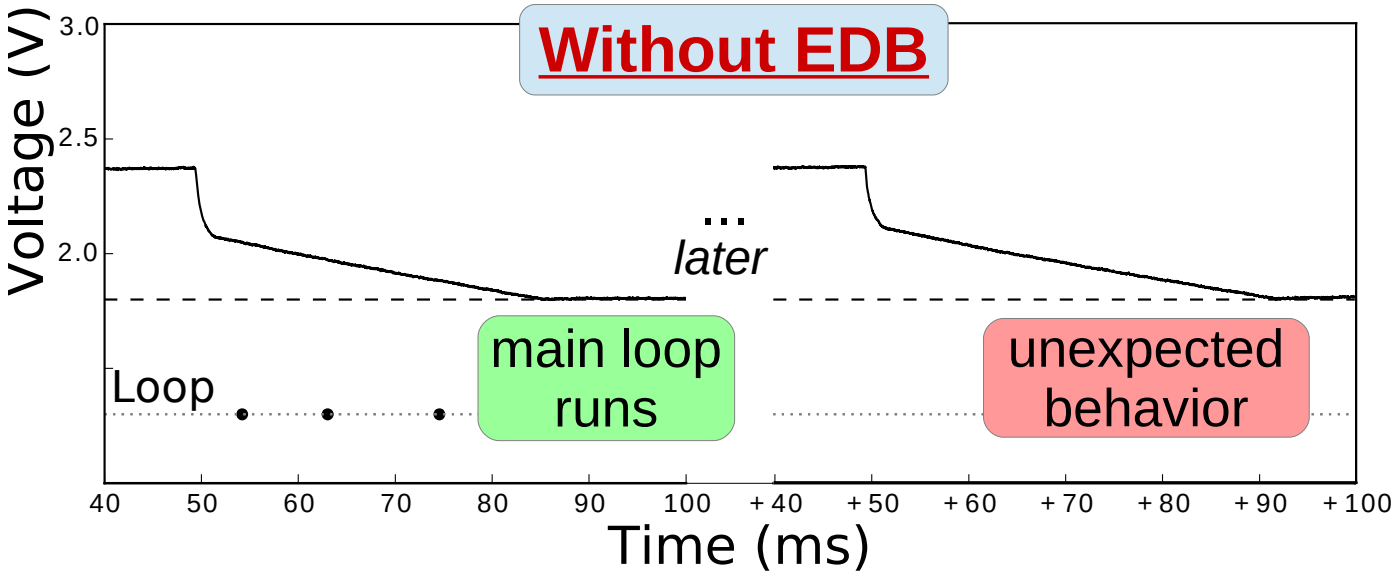
```
main:

 while True:
   append(...)
```



Evaluation Case-Study (II): Keep-Alive Assertions

broken_fibonacci.c

```
main:

 while True:
   append(...)
```

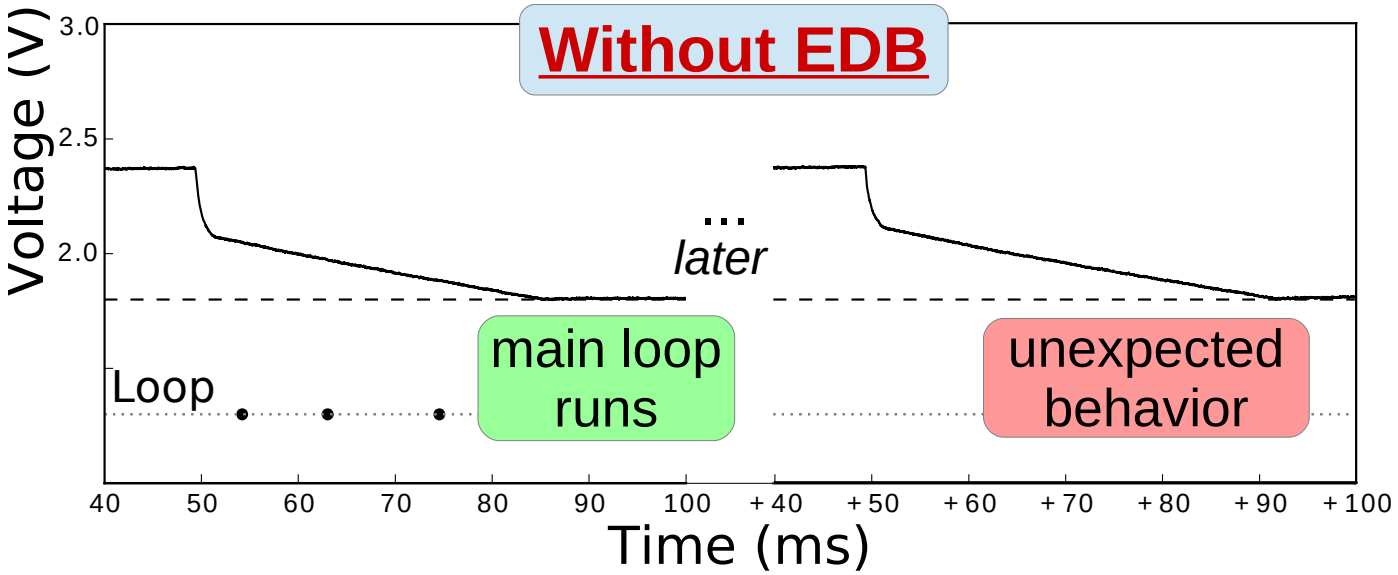


Evaluation Case-Study (II): Keep-Alive Assertions

broken_fibonacci.c

```
main:

 while True:
   append(...)
   assert(...)
```

Diagnosis strategy



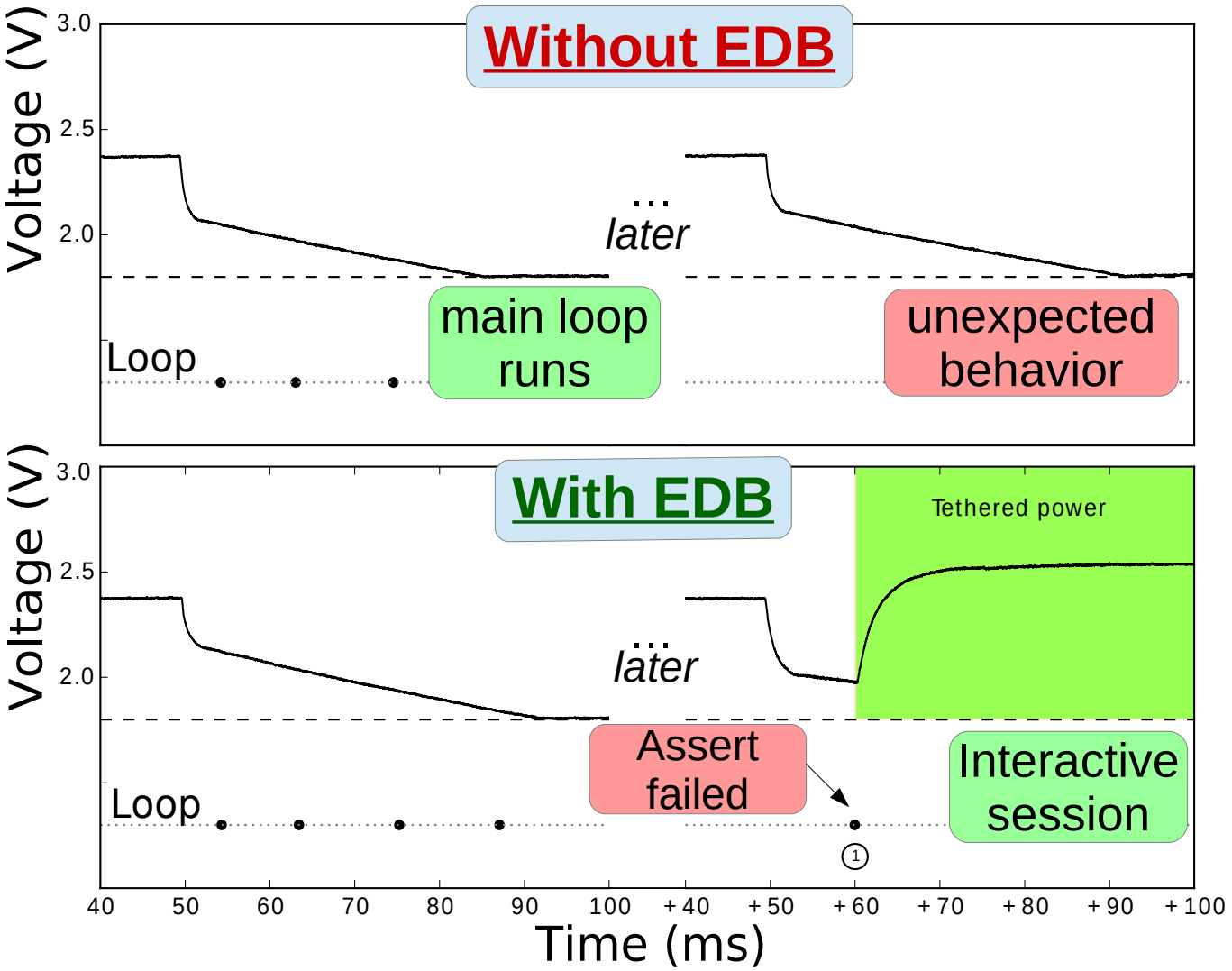
Evaluation Case-Study (II): Keep-Alive Assertions

broken_fibonacci.c

```
main:

 while True:
   append(...)
   assert(...)
```

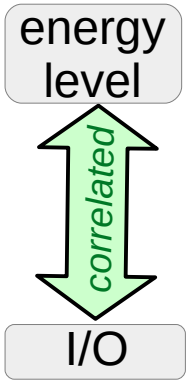
Diagnosis strategy

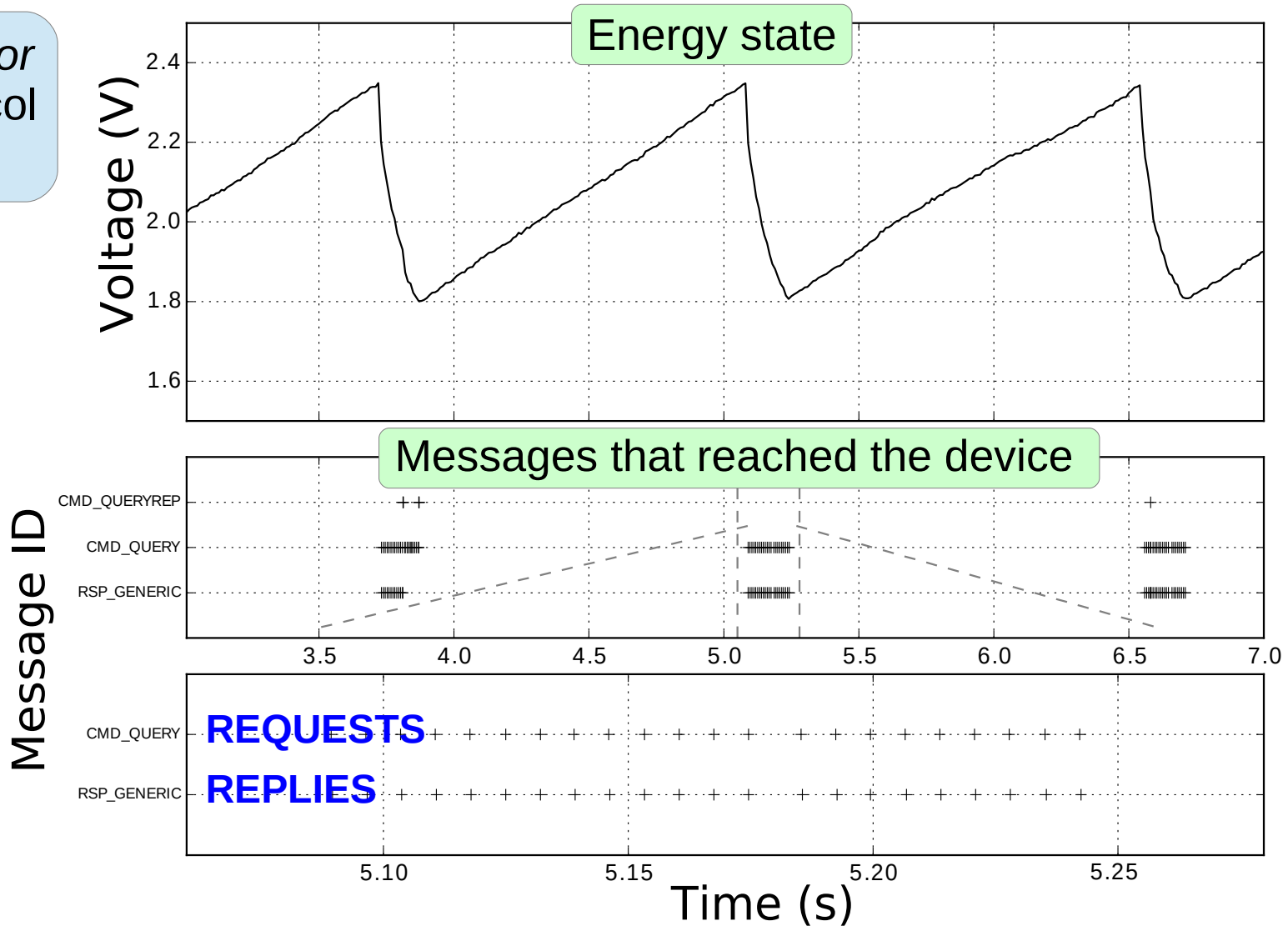
EDB *keep-alive assert* brings the familiar primitive to energy-harvesting platforms.



Evaluation Case-Study (III): Monitoring of RFID I/O

EDB *I/O monitor* enables protocol debugging.

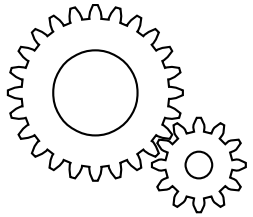




Evaluation Roadmap

- Case studies: applying EDB to debugging tasks

  (I) Zero-cost instrumentation
  (II) Keep-alive assertions
  (III) I/O monitoring



- Quantifying energy-interference

  - Leakage current
  - Save-restore resolution

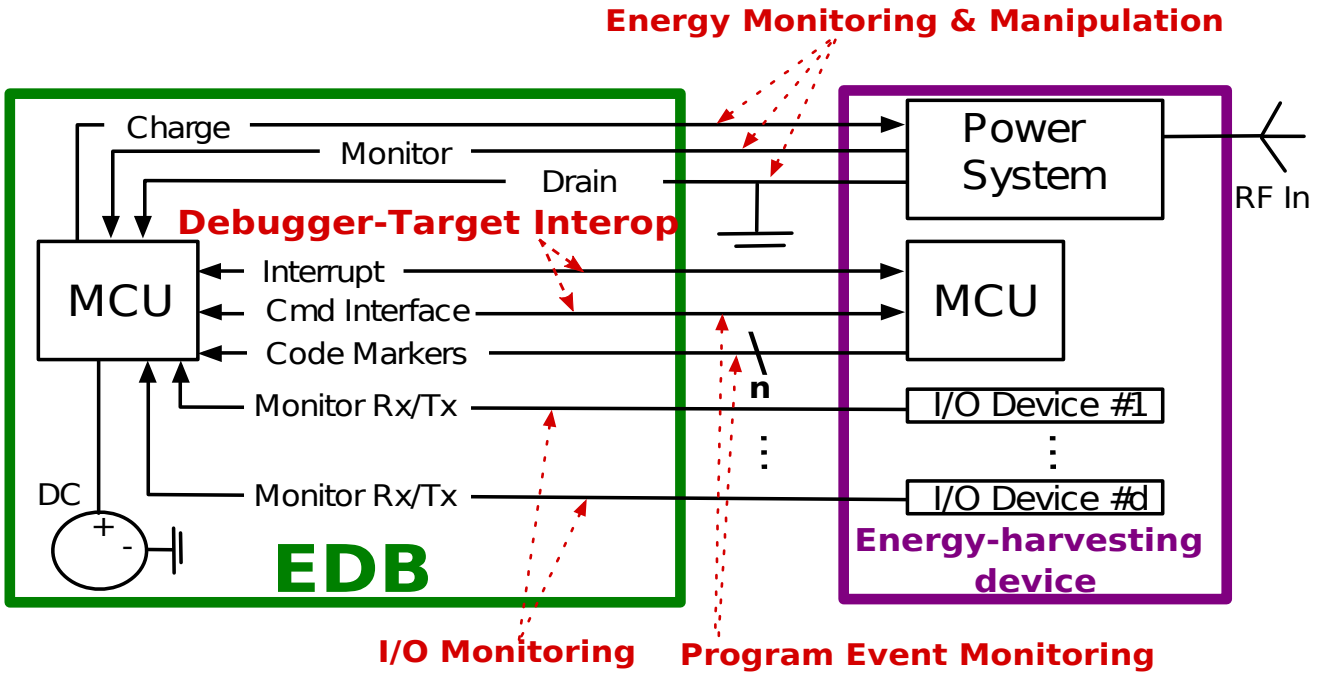
Evaluation: Energy-interference: Worst-case Leakage Current

$I_{leak} \leq 850$ nA

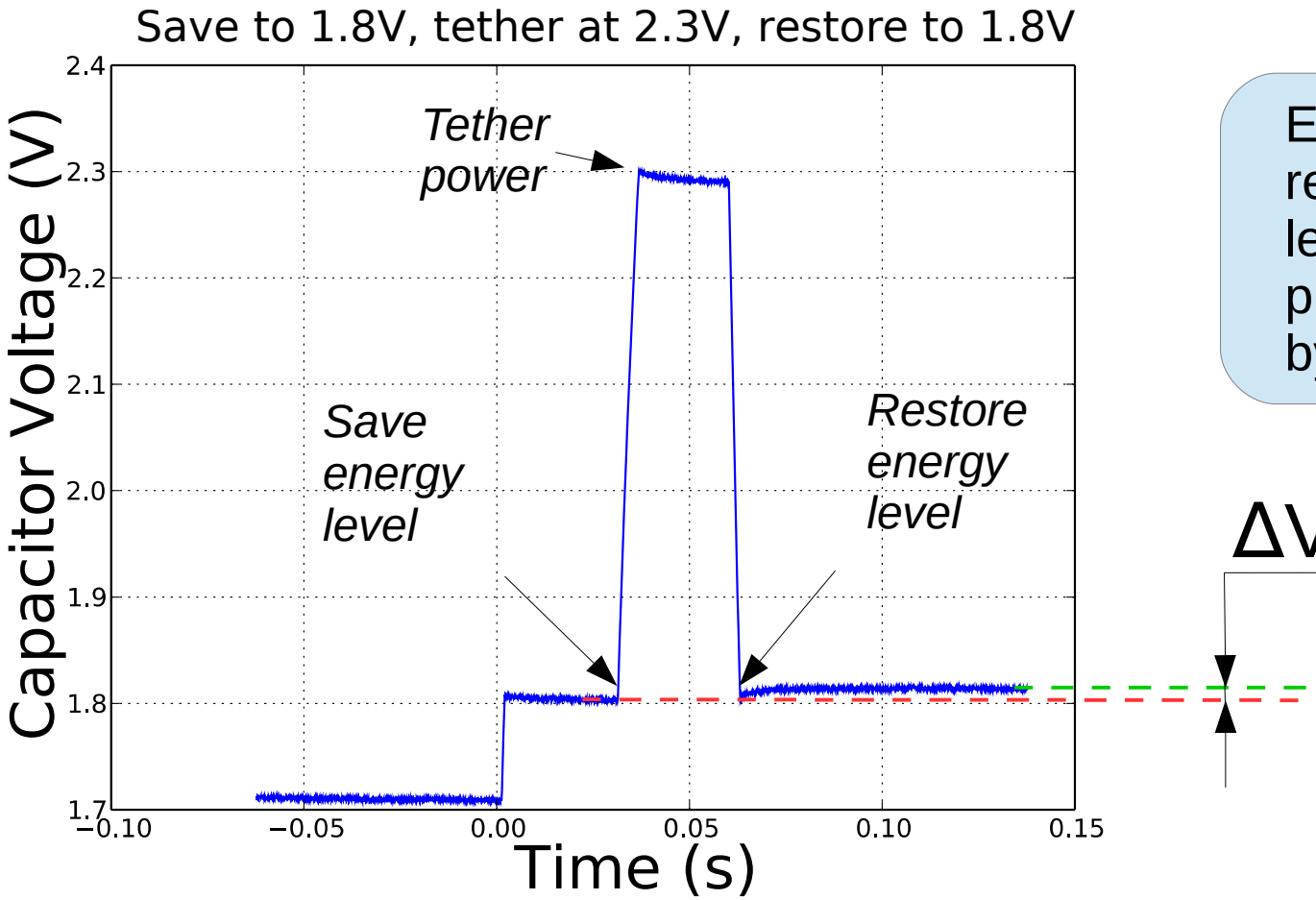
$\simeq$ 0.4% of min. MCU current

Sources of leakage current

Energy Level Sense & Set	Program Event Monitoring	EDB-Target Interop	I/O Monitoring



Evaluation: Energy-interference: Save-Restore Accuracy



EDB saves and restores energy level with a finite precision, limited by hardware.

ΔV = 54 mV ≋ 4% of energy capacity on WISP device (47µF)

Related Work

- Intermittent computing

  DINO [B. Lucia, B. Ransford. PLDI '15]

  Mementos [B. Ransford, J. Sorber, K. Fu. ASPLOS '11]

- Emulation and simulation of energy sources

  Ekho [J. Hester, T. Scott, J. Sorber. SenSys ’14]

  CRFID Crash Test Sim [J. Gummeson, S. Clark, K. Fu, D. Ganesan. MobiSys ’10]

- Debugging battery-powered wireless sensor nodes

  Clairvoyant [J. Yang, M. L. Soffa, L. Selavo, K. Whitehouse. SenSys ’07]

Conclusion

Energy-interference-free Debugger (EDB) is the first system to bring advanced debugging capabilities to intermittent energy-harvesting platforms.

http://intermittent.systems/

An Energy-interference-free Hardware-Software Debugger for Intermittent Energy-harvesting Systems

Alexei Colin*^ Graham Harvey*
Alanson Sample* Brandon Lucia^

*Disney Research Pittsburgh
^Dept. of ECE, Carnegie Mellon University

Carnegie Mellon University
Electrical & Computer ENGINEERING

Disney Research, Pittsburgh