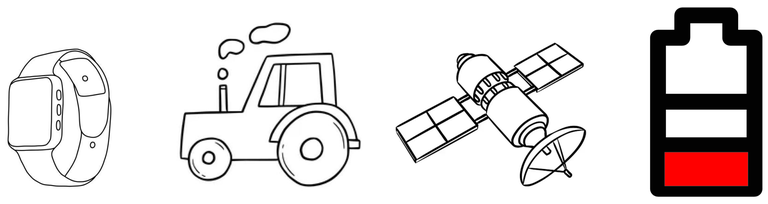
Automatic Energy Analysis Using Types

Sai Divvela

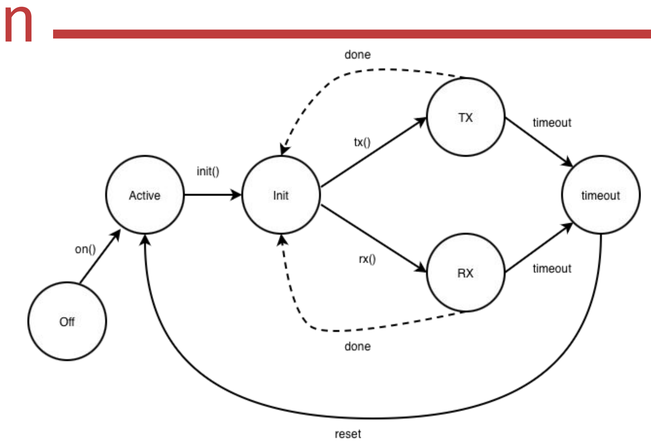
UNIVERSITY OF MARYLAND

PLUM

Background + Motivation



Embedded systems have limited energy budget - used in diverse applications



Complex state difficult to reason about

Two prior approaches: measurements, static analysis

Measurement	Static Analysis
Accurate but difficult to do	Easy but relies on assumptions

Our Approach

Combine automatic amortized resource analysis with **typestate** for accurate, expressive peripheral reasoning

Categories of energy costs

1. Main board execution
2. Peripheral operations
3. Active **background** peripherals

`some_other_func()` represents generic computation

Code	I2C State
`transition(i2C, Tx);`	? → Tx
`i2C.transmit();`	Tx
`some_other_func();`	Tx
`i2C.stop();`	Tx → Stop

Cost Reporting

- C - fixed expression cost
- C(p : st) - peripheral p in st
- C(p : (st, st')) - peripheral p transitioning st -> st'

Symbolic costs for hardware agnostic reasoning

Cost Analysis Example

```
fn log_data(temp) -> u64 {
 transition(temp, Mes);
 let res = temp.measure();
 let t = 0;
 if(res <= 65) {
  t = 1;
 }
 transition(temp, Idle);
 return t;
}
```

C(temp: (???, Measure))
C_m + C(temp: Measure)
C_v + C(temp: Measure)

max(C_v, 0) + C(temp: Measure)

C(temp: (Measure, Idle))

Total Cost: C_l = C(temp: (???, Measure)) + 2C_v + C_m + 3C(temp: Measure) + C(temp: (Measure, Idle))

State Inference Motivation

Initial state depends on previous behavior

Annotations can be error-prone
Polymorphism can lead to conservatism

```
fn main() {
 transition(temp, Idle);
 let t = log_data(temp);
}
```

C(temp: (Off, Idle))

C_l[?? = Idle]

```
fn main() {
 transition(temp, Idle);
 transition(temp, Measure);
 let m = temp.measure();
 let t = log_data(temp);
}
```

C(temp: (Off, Idle))
C(temp: (Idle, Measure))
C_m + C(temp: Measure)
C_l[?? = Measure]

State Inference Example

New state variable per transition - similar to type variables

```
fn main() {                          temp = [α0]
 transition(temp, Idle);             temp = [α1 :: α0]
                                     Φ = α0 ⩽ α1 ∧ α1 = Idle
 let t = log_data(temp);             solve(Φ ∧
}                                          α1 ⩽ Measure)
```

Constraints combined at function calls

State variable stack tracks peripheral history

Conclusion

Type system for reasoning about energy consumption with stateful peripherals.
Next steps: support async operations, finish implementation