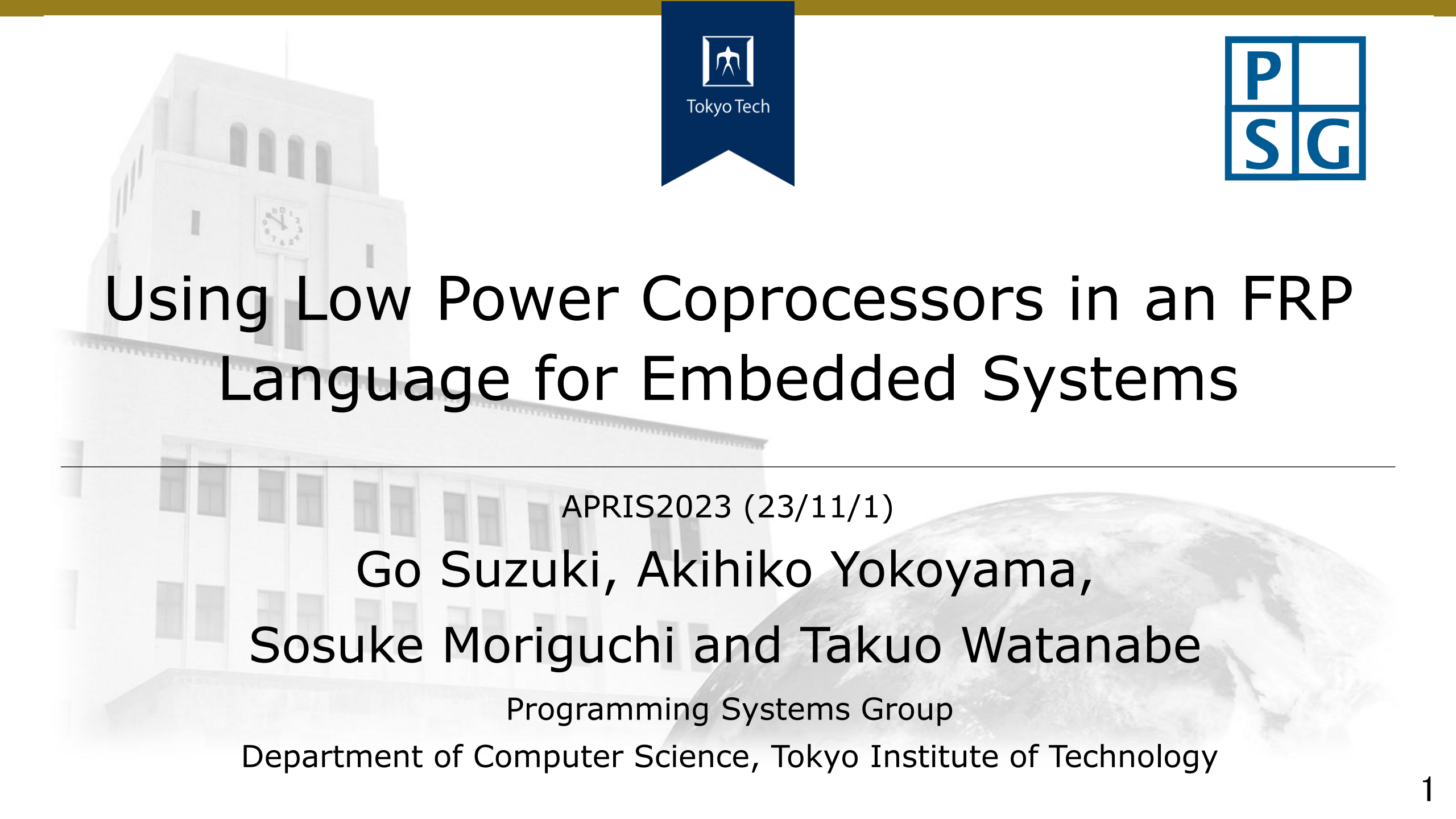


Using Low Power Coprocessors in an FRP Language for Embedded Systems

APRIS2023 (23/11/1)

Go Suzuki, Akihiko Yokoyama,
Sosuke Moriguchi and Takuo Watanabe

Programming Systems Group
Department of Computer Science, Tokyo Institute of Technology

About This Work

Goal	To provide a good abstraction mechanism for using low power coprocessors to reduce power consumption of embedded systems.
Approach	Introduce a mechanism for switching running processors to XStorm, an FRP language for embedded systems.
Evaluation	Observed lower power consumption and acceptable state transition time overhead.

Backgrounds

1. Functional Reactive Programming (FRP)

A) A Theremin Example

B) Key concept of FRP: Time-Varying Values

C) XStorm, an FRP language with an abstraction mechanism for modeling stateful behaviors.

2. Low Power Coprocessor

A) Target : ESP32-S3 and RISC-V Ultra Low Power Coprocessor (ULP Coprocessor)

B) An example to reduce power consumption

Example : Theremin

"Theremin" : an electronic musical instrument whose sound varies according to the position of the performer's hands.

Example Implementation

Distance Sensor (ToF) for frequency

Button for volume

I²C

GPIO

Microcontroller (ESP32-S3)

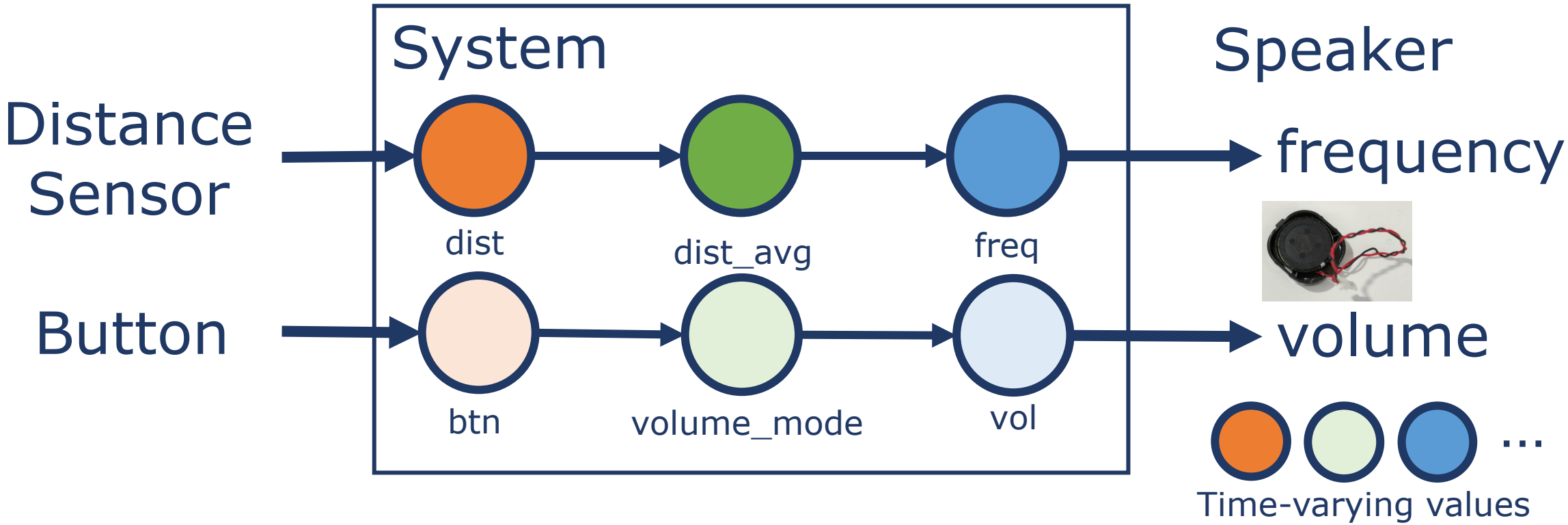
PWM

Speaker

Functional Reactive Programming (FRP)

Time-Varying Values (aka **Signals**)

abstraction of values that change over time



XStorm [Matsumura et al. 2020]

An FRP Language for small-scale embedded devices
(AVR, ESP32, Cortex-M, etc.)

Translates to Standard C (or C++).

Features

- **An abstraction mechanism for stateful behaviors**
- Statically-typed
- Simple Syntax
  (no lifting, no callbacks)
- **Statically-determined runtime memory size**

Theremin in XStorm

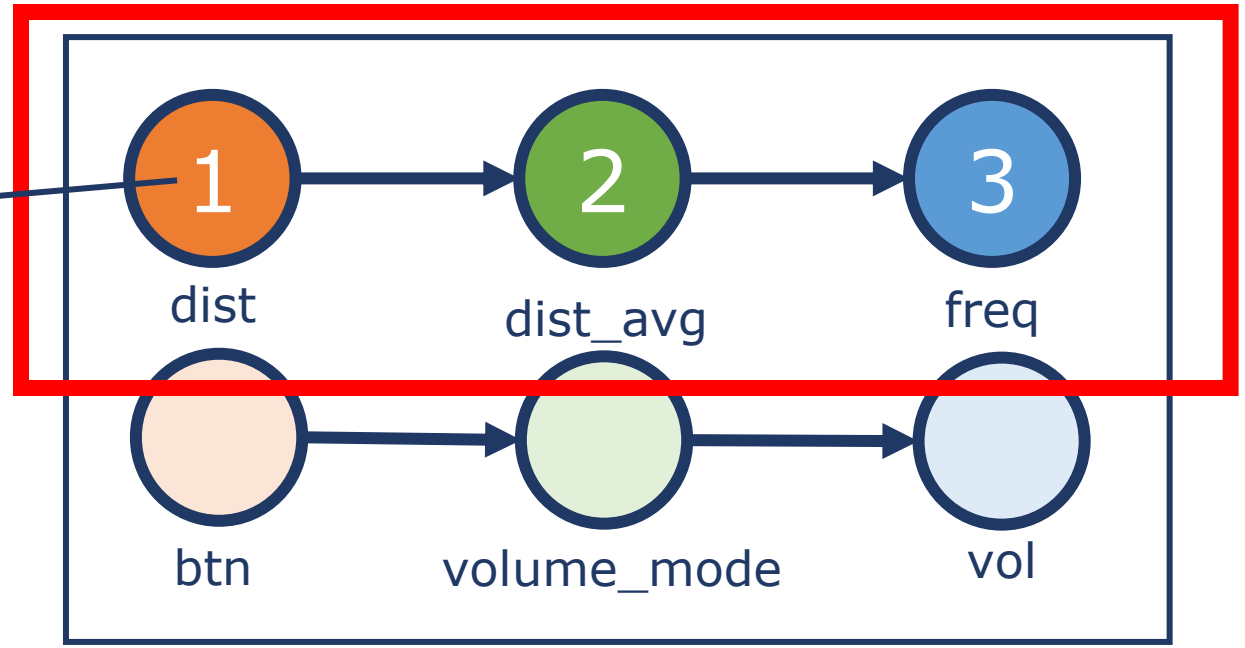
Node : time-varying value in XStorm

```
node dist_avg = (dist * 4 + dist_avg@last * 6) / 10
node freq = dist_avg * 2
```

`dist_avg` is a moving average of `dist`.

Iteration: Each node is updated according to the dependency.

`dist` is updated before `dist_avg`.



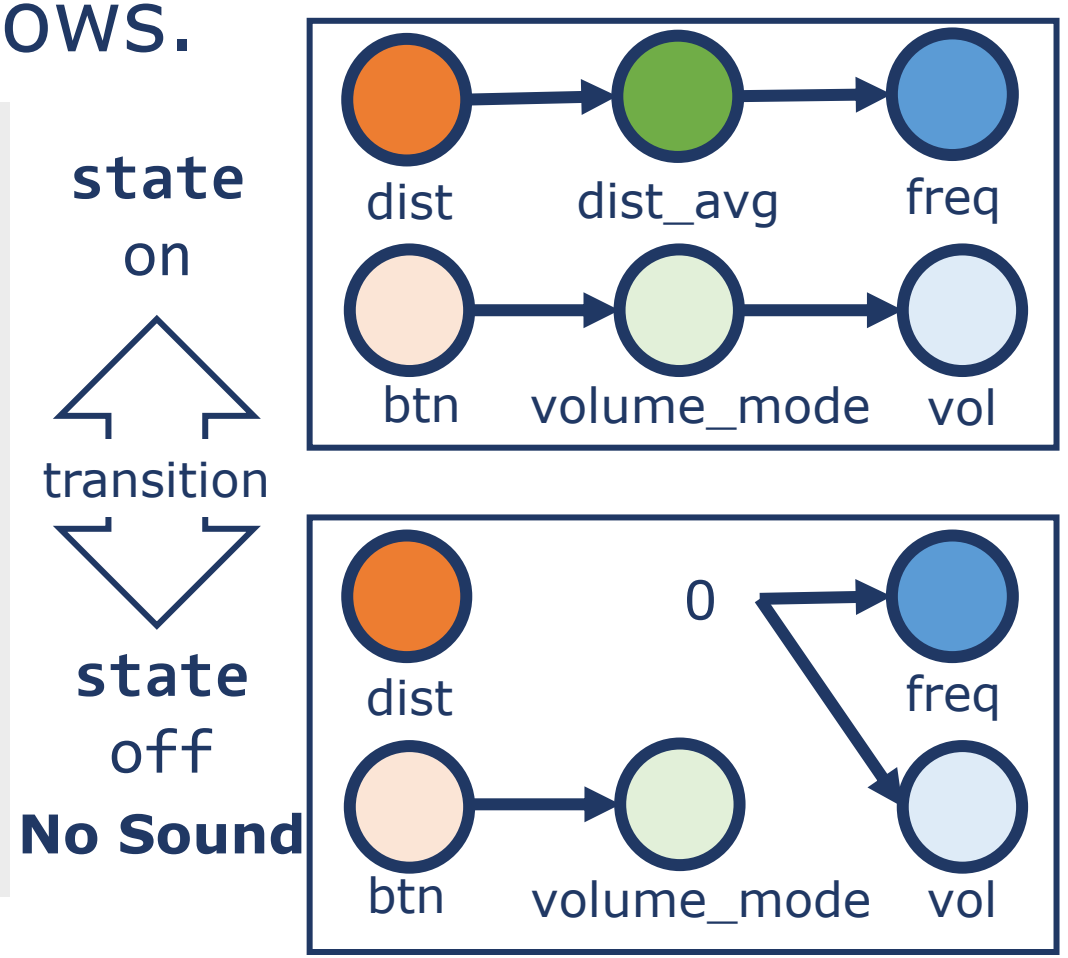
The generated C program repeats iterations for the reactivity.

Theremin in XStorm : Stateful Behavior

The Theremin has two states. They can be described in XStorm as follows.

```
state On {
  node dist_avg = (dist * 4 …
  node freq = dist_avg * 2
  …
}

state Off {
  node freq = 0
  …
}
```

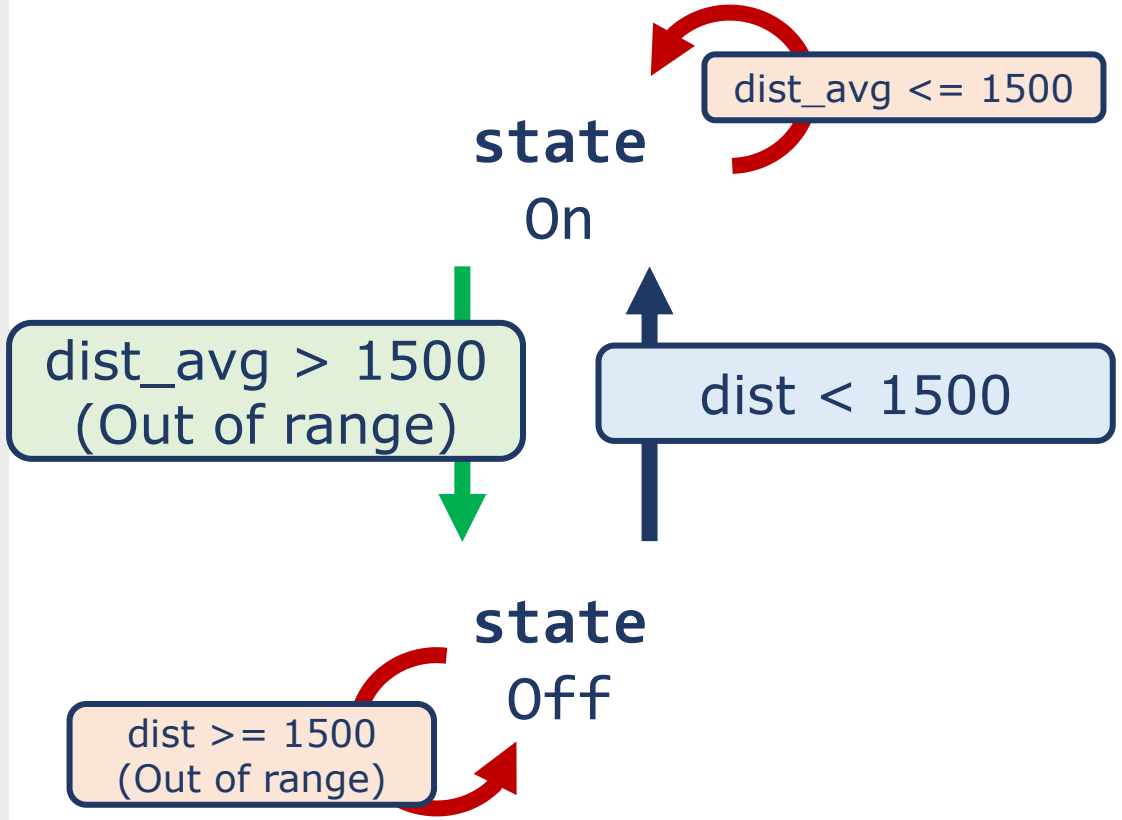


Theremin in XStorm : Stateful Behavior

`switch` specifies the next state. `Retain` represents staying in the current state.

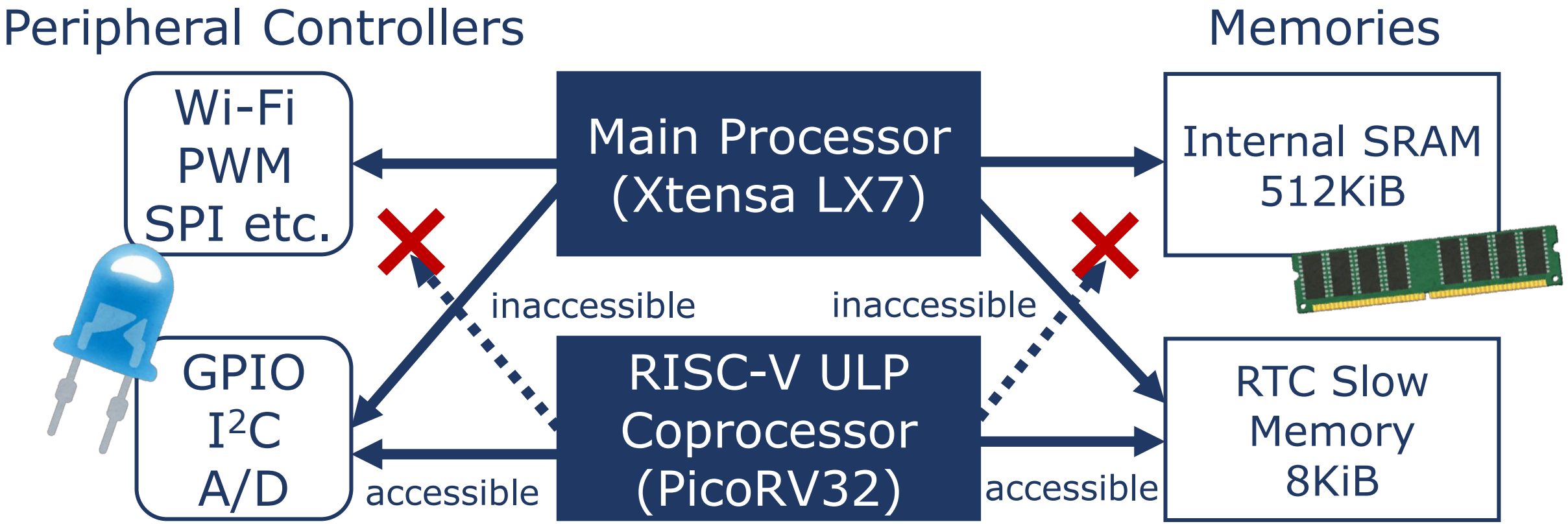
```
state On {
  node …
  switch: if dist_avg > 1500
            then Off else Retain
}

state Off {
  node …
  switch: if dist < 1500
            then On else Retain
}
```



Target: ESP32-S3

ESP32-S3 has RISC-V Ultra Low Power coprocessor (ULP coprocessor).



Sleep States

Two Sleep states: **Light-Sleep** and **Deep-Sleep** can reduce power consumption. **The ULP coprocessor runs under these sleep states.**

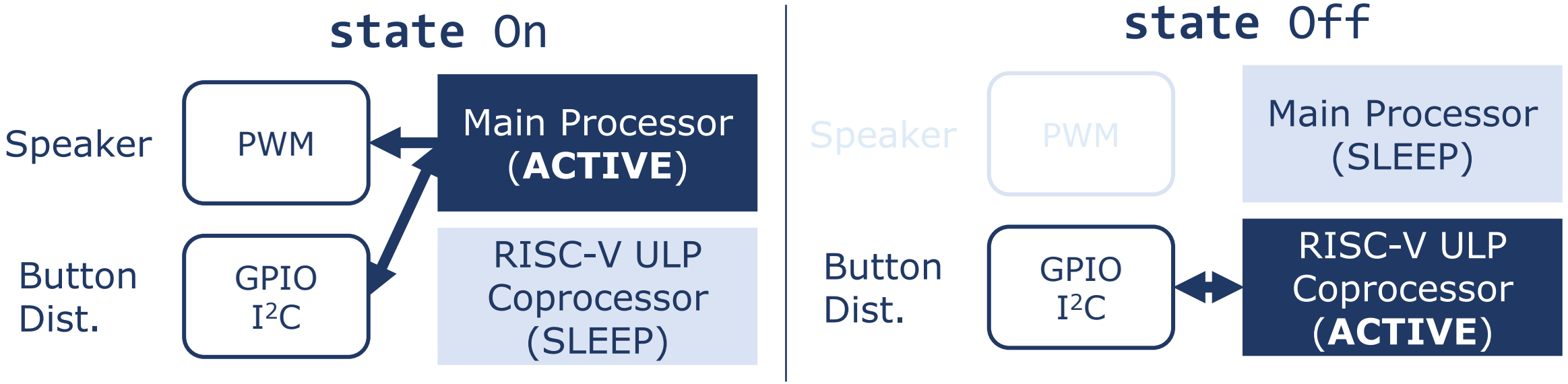
	Active (Modem-Sleep, 160MHz)	**Light-Sleep**	**Deep-Sleep**
Internal SRAM		● Retain	× Discard
Program Counter of the Main Processor		● Retain	× Return to the entry point
Typical Consumptions	× 39.9 mA	240 μA	● 7 μA

Typical consumptions (3.3V) from ESP32-S3 Series Datasheet v1.7 (2023-06)
Active with lowest frequency (40MHz) consumes 13.2 mA

An Example to Reduce Power Consumption

Maintaining the reactivity in the state Off with the ULP coprocessor can reduce power consumption.

The ULP coprocessor supports I²C, so interaction by the distance sensor is maintained.



Advantage/Disadvantage of the ULP Coprocessor

- Advantage : Lower power consumption than the main processor. 7 μA (ULP) vs. 13 mA (lowest freq.)
- Disadvantages
  - ✓ Different ISA & I/O configuration
    - Limited functionality (e.g., no FPU) leads to lower power consumption, but more difficult to use (Execution migration solutions are expensive.[Q&A Page])
  - ✓ Limited memory space and different memory mapping
  - ✓ Need for the processor power state management
  - ■ Higher power consumption with frequent switching (v.s. DVFS)

Proposed Mechanism

We introduce **"on" modifier** to XStorm, which declares which processor to run in each state.

```
state On on main {
  node …
  switch: …
}

state Off on ulp {
  node …
  switch: …
}
```

The compiler produces:

- A program running on the main processor.
- Data transfer between processors.
- A program running on the ULP coprocessor.

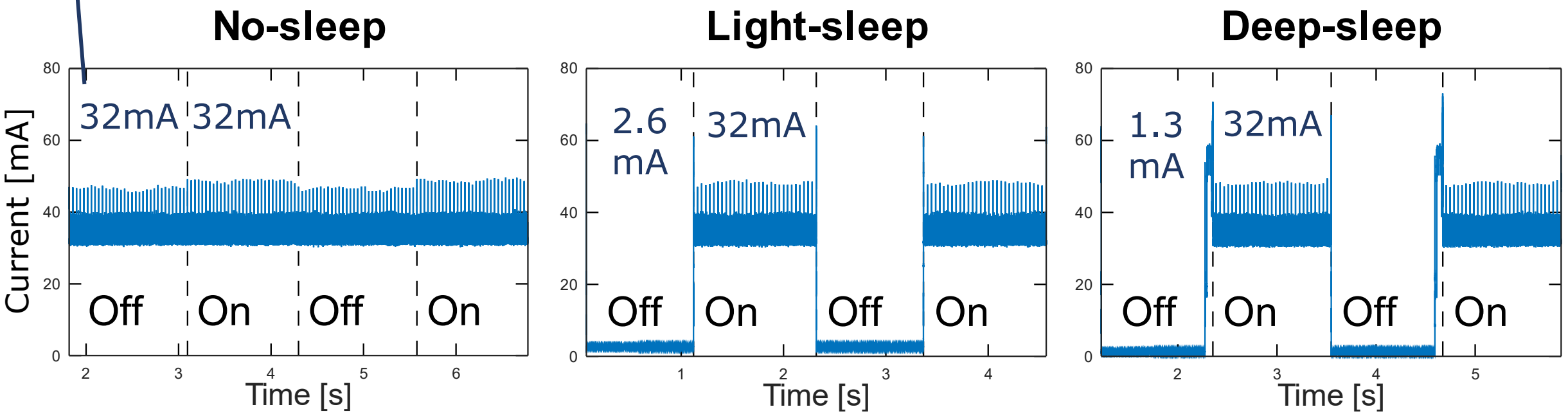
Evaluation

- Evaluations using the Theremin example:
  - Power consumption (excluding peripherals)
    - Comparison: No Coprocessor, Light-Sleep, Deep-Sleep
  - Latency of state transitions
    - Comparison: Light-Sleep, Deep-Sleep
- Environment:
  - ESP32-S3-DevkitC-1 N8 v1.0 board
  - Ammeter (Power Consumption): Nordic Power Profiler Kit 2 (3.3V output)
  - FPGA (Latency Measurement): ZYBO Z7-20 (50MHz)

Result: Power Consumption

We observed that, in the state `Off`, the current is lower than in the state `On` with the ULP coprocessor.

Mean of 10th ~ 90th percentiles



Result: Latency

We measured the overhead of state transitions.

	Light-Sleep	Deep-Sleep
Data Transfer Main -> ULP	9.63	13.91
Data Transfer ULP -> Main	14.64	39.43
ULP Wake-up	164.00	179.75
Main Wake-up	579.29	79669.60

[μs]

Data Transfer takes less time than processors' wake-ups.

Boot Process consumes a lot of time.

Related Work

Other FRP languages for embedded systems

Hailstorm [Sarkar, A. et al '20] and Juniper [Helbling, C. et al '16] have different ways of describing stateful behaviors.

```
Signal:foldP(fn (inputs, state) ->
   case state of
   | On => ( ... ) // state On { ... }
   | Off => ( ... ) // state Off { ... }
   end
end, inputSig)
```

Related Work: Other microcontrollers

- There are several microcontrollers with low power coprocessors.
- Our prototype is specialized for ESP32-S3, but the adaption is not difficult for the microcontrollers that (e.g., i.MX RT, LPC and PSoC):
  - have Inter-communication via memory-mapped I/O
  - have C API to enter the sleep state
  - restart from the entry point or the last position
  - can access before coprocessor wake-ups or the coprocessor can wait for data transfer

Summary

- Our proposed mechanism allows us to use the ULP coprocessor and reduce power consumption.
  - An abstraction mechanism to describe stateful behaviors with a low power coprocessor
  - Data transfer is automatically generated by the compiler
- The evaluations show that our prototype reduces power consumption and data transfer is acceptable overhead in ESP32-S3.
- Future Work: Support for other microcontrollers.

www.psg.c.titech.ac.jp