



**HAL**  
open science

## Memory Allocation in Intermittent Computing

Hugo Reymond, Isabelle Puaut, Erven Rohou, Sébastien Faucou, Jean-Luc Béchenec, Mikaël Briday

► **To cite this version:**

Hugo Reymond, Isabelle Puaut, Erven Rohou, Sébastien Faucou, Jean-Luc Béchenec, et al.. Memory Allocation in Intermittent Computing. COMPAS'2022, Jul 2022, Amiens, France. hal-04385204

**HAL Id: hal-04385204**

**<https://hal.science/hal-04385204v1>**

Submitted on 10 Jan 2024

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

## 1. CONTEXT

**Battery-free** sensing devices can operate in inaccessible locations (space, civil buildings) without the need of maintenance. They are equipped with the following:

- An **energy harvester** (solar panel, piezogenerator ...) that harvest energy from its environment
- An **energy buffer** (supercapacitor) to buffer the small amount of energy harvested
- A normally-off **microcontroller**, executing short burst of computations with the buffered energy

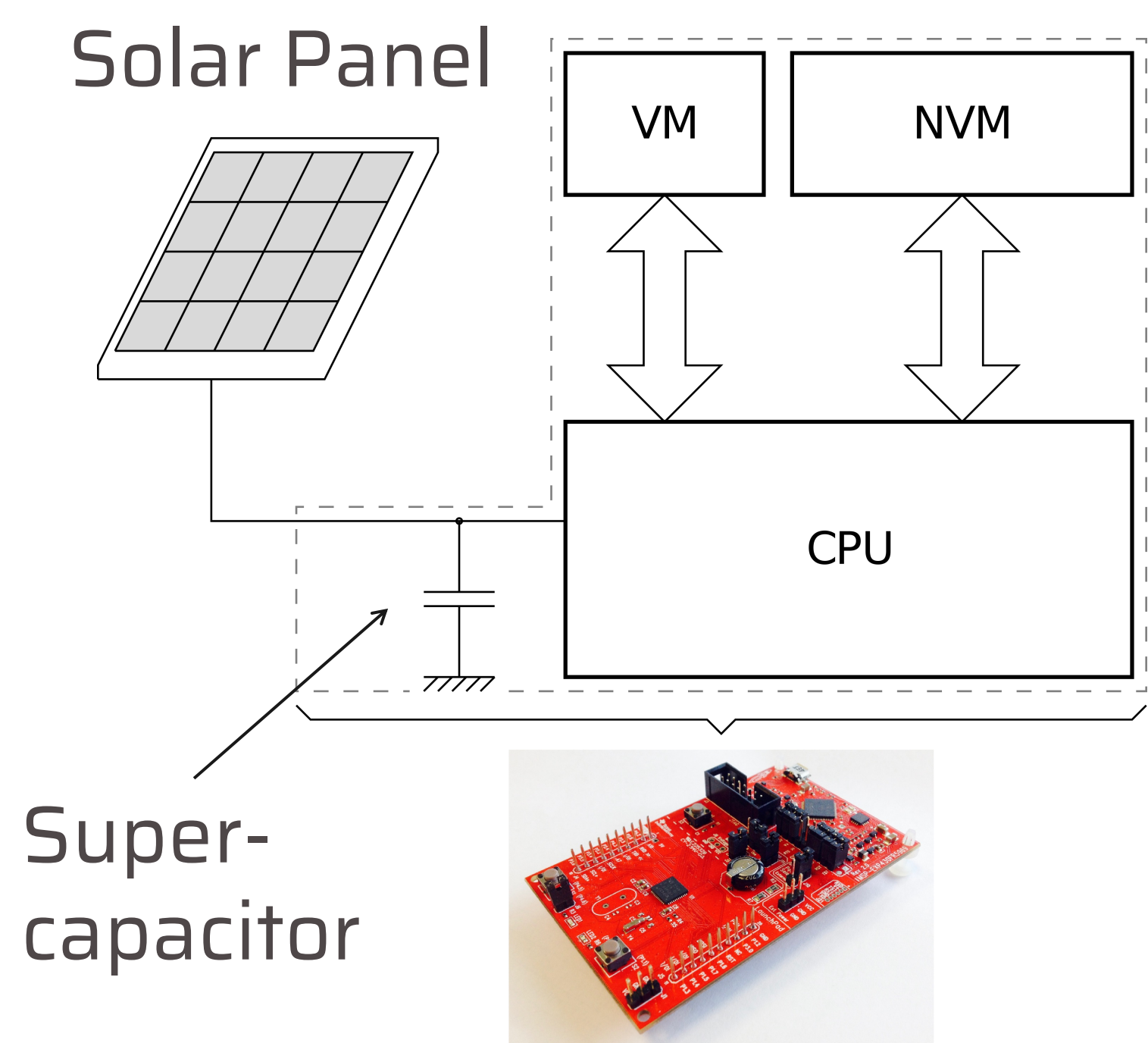


Figure 1 Architecture of an intermittent system

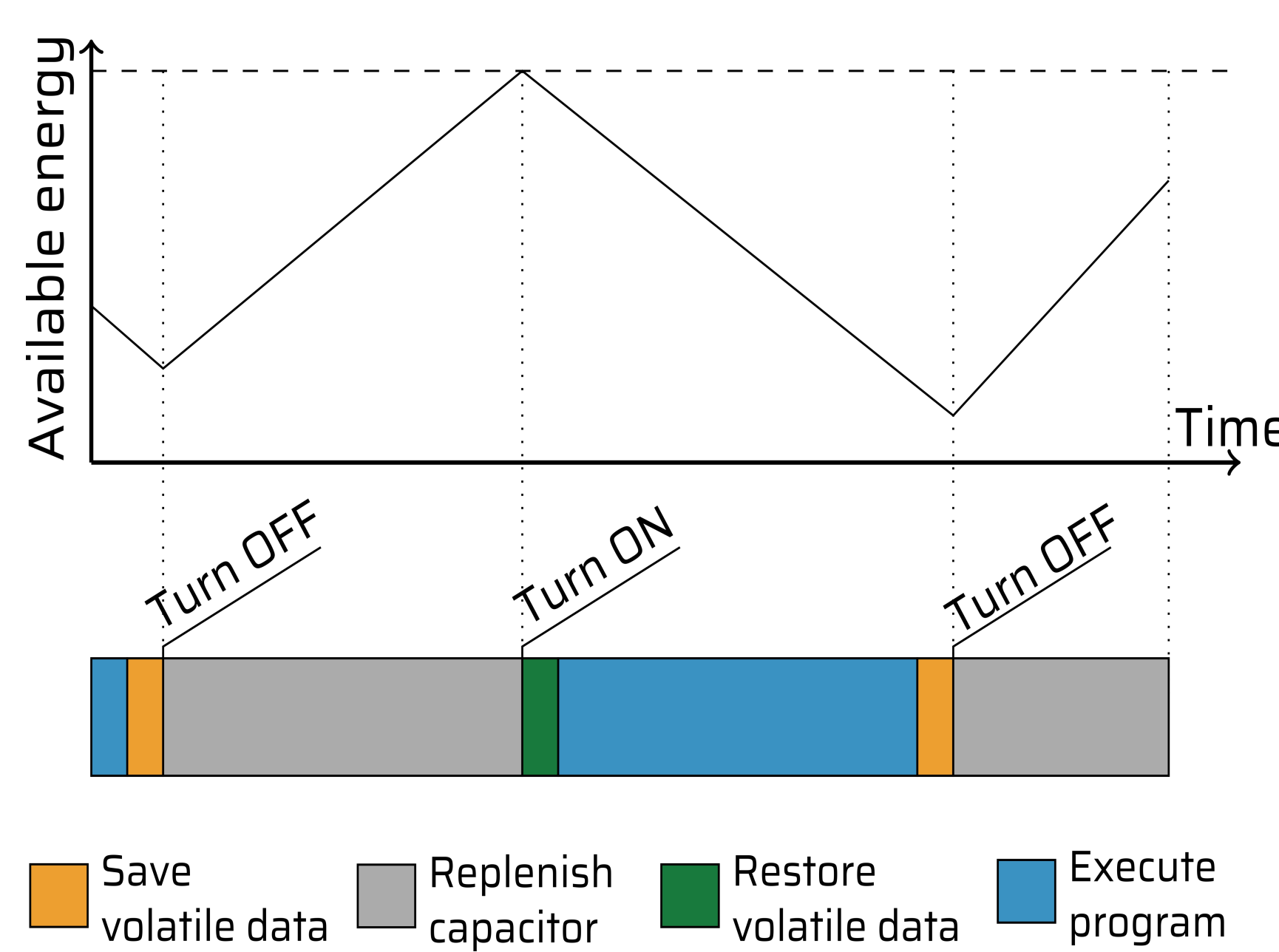


Figure 2 Intermittent execution

## 2. CHECKPOINT PLACEMENT AND MEMORY ALLOCATION

**Problem:** progress is lost on power off since volatile memory (VM) is erased

→ **Solution:** Insert instructions in the program binary to save checkpoints in non-volatile memory (NVM)

↳ Where?

Should variable be stored in VM or in NVM?

↳ In VM -> Low energy accesses (2.5x less than NVM accesses)

↳ In NVM -> No checkpointing overhead

```

int sum = 0; // Save checkpoint here?
for(int i = 0; i < SIZE; i++)
    sum += array[i]; // Save checkpoint here?
/* ..... */ // Save checkpoint here?
int a = f(sum); // Store in VM or NVM?
    
```

## 3. OUR SOLUTION: SCHEMATIC

**Our goals:**

- Take into account volatile memory size
- Ensure program termination
- Reduce energy consumption

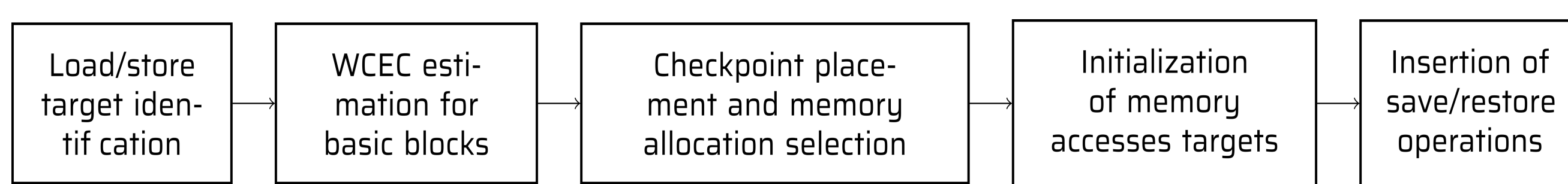


Figure 3 SCHEMATIC pipeline

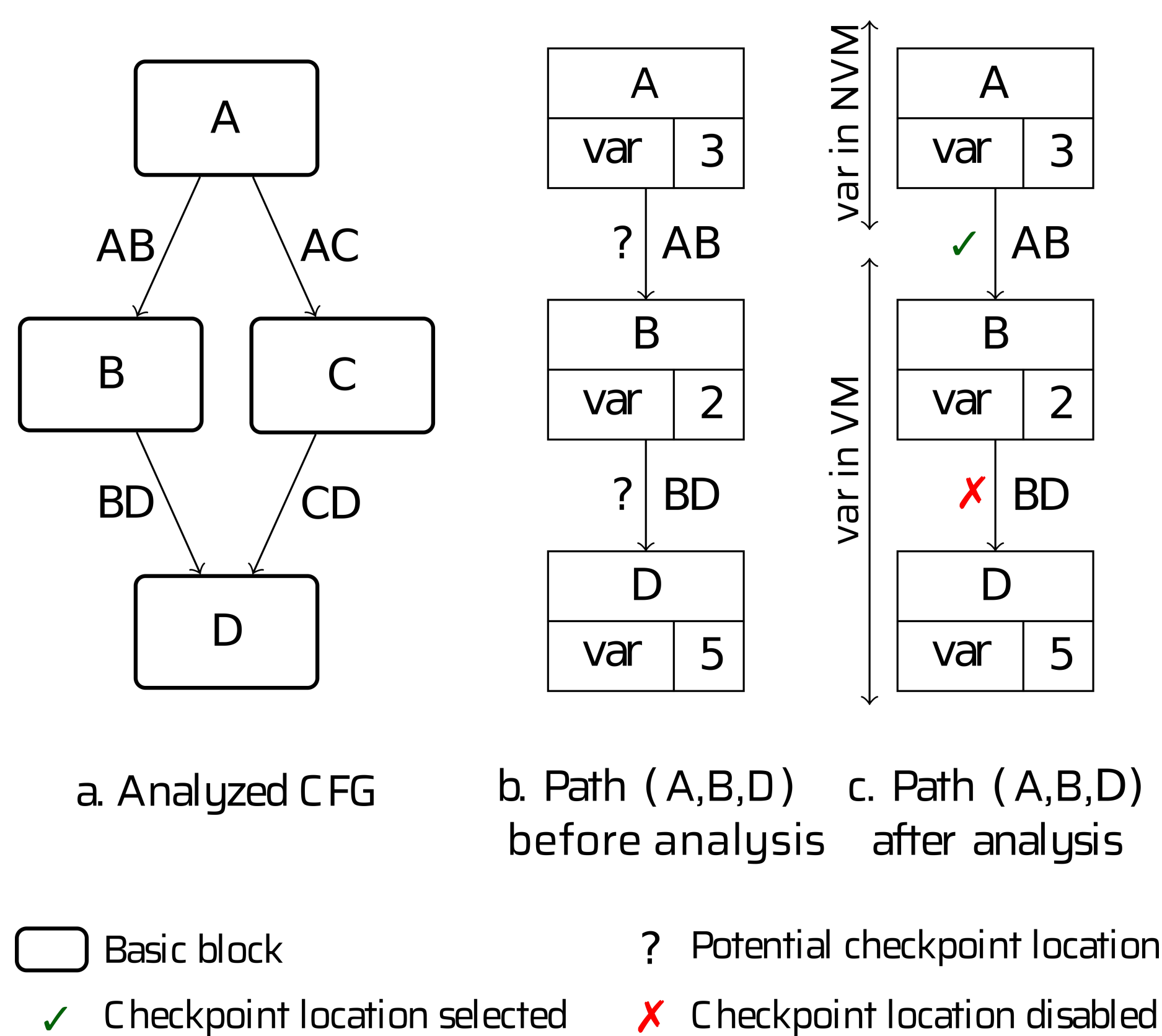


Figure 4 Analysis of a program using its CFG representation

## 4. RESULTS

	Benchmarks (MiBench Suite)		
	CRC	AES	FFT
Ratchet (all-NVM)	✓	✓	✓
Mementos (all-VM)	✓	✓	✗
Alfred	✓	✓	✗
SCHEMATIC	✓	✓	✓

Figure 5 Ability of solutions to run on a microcontroller with a limited volatile memory (MSP430FR5969: 2kB VM, 64kB NVM)

**1.68x** less energy consumed with our solution on average

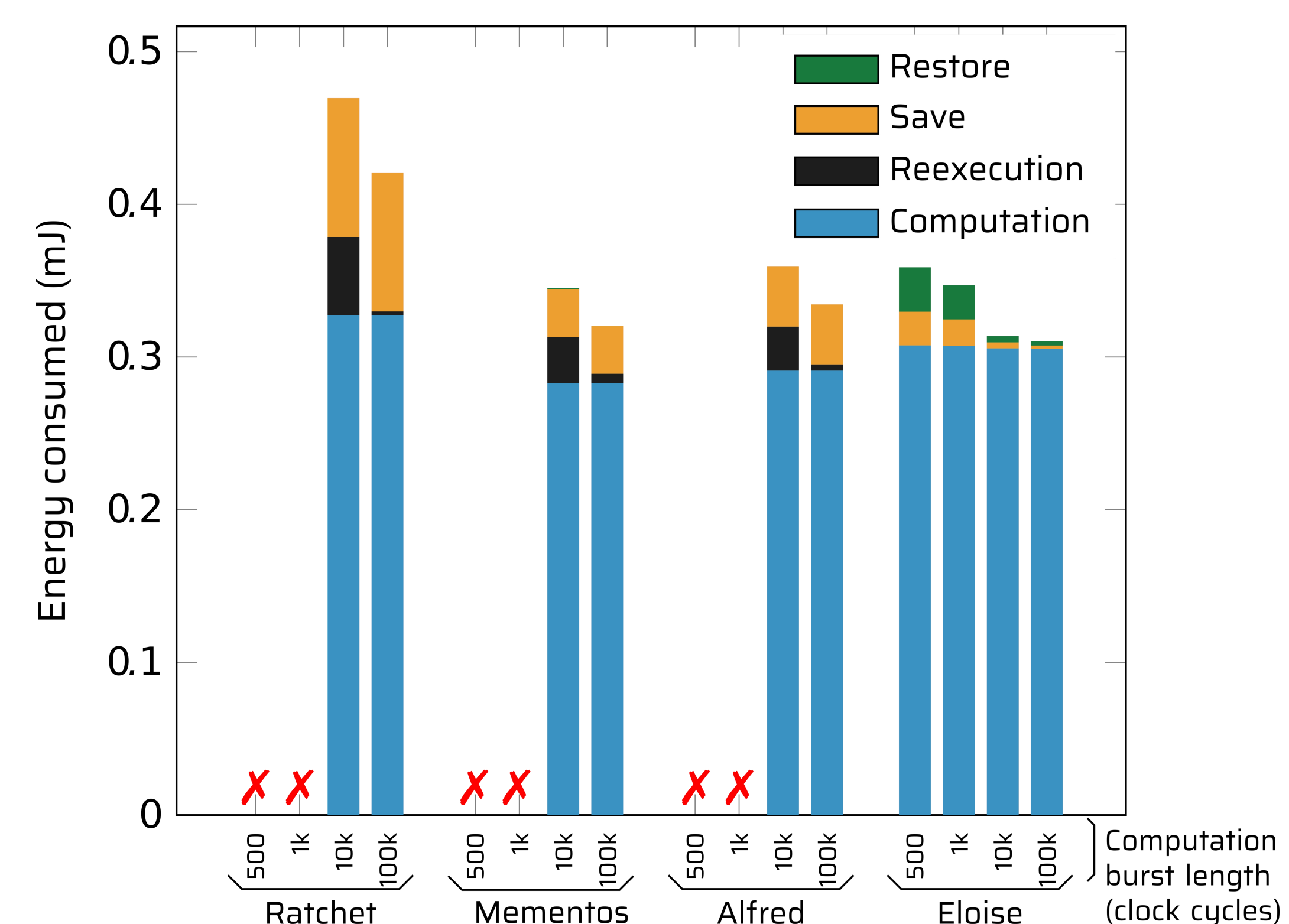


Figure 6 Energy consumption of SCHEMATIC in comparison to other solutions. ✗ means that the baseline could not execute

### Bibliographie / sources

- (1) Woude, J. V. D.; Hicks, M. "Ratchet:" Intermittent Computation without Hardware Support or Programmer Intervention; OSDI'16; 2016;
- (2) Ransford, B.; Sorber, J.; Fu, K. Mementos: System Support for Long-Running Computation on RFID-Scale Devices; ASPLOS XVI; 2011-
- (3) Maioli, A.; Mottola, L. ALFRED: Virtual Memory for Intermittent Computing. SenSys '21; 2021
- (4) Reymond, H & al. SCHEMATIC: Compile-time Checkpoint Placement and Memory Allocation for Intermittent Systems. CGO'24; 2024