# Let’s Meet 12/10/2021

## Computer Science @Bruges Campus

## Embedded Software Resilience

dr. ing. Jens Vankeirsbilck

IEEE Student Branch KU Leuven Campus Brugge

m·group

# Research
# @ Computer Science Bruges Campus

- Part of M-Group → focus on Connected Mechatronic Systems
  - Ultimate machine
  - Ultimate factory
- Focus on Dependability and Safety
  - Reliable communication (Bozheng, Kristof, Pejman)
  - Smart condition monitoring (Chandu, Jens D., Paul)
  - Certain Artificial Intelligence (Calvin, Keivan)
  - CoMoveIt (spin-off) (Sotirios)
  - Resilient embedded systems (Mohaddaseh, Brent, Jens Vkb)

- prof. Jeroen Boydens
- prof. Hans Hallez
- prof. Mathias Verbeke

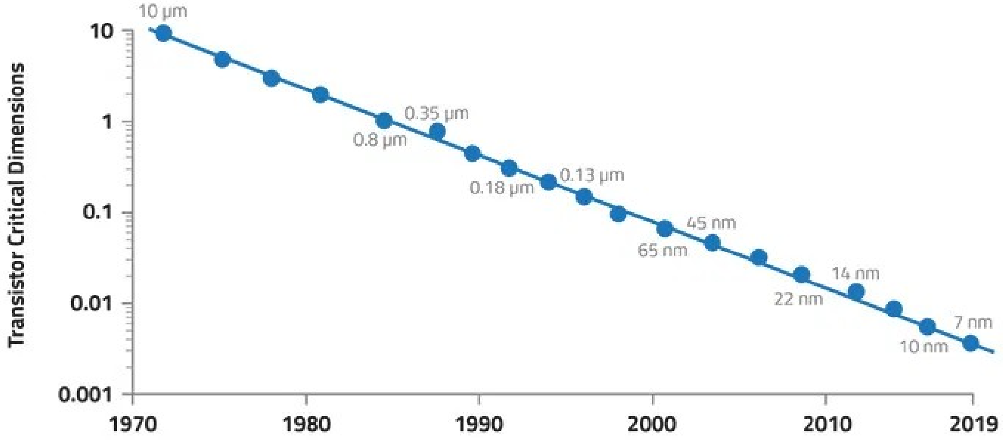
# Embedded Software Resilience

# Overview

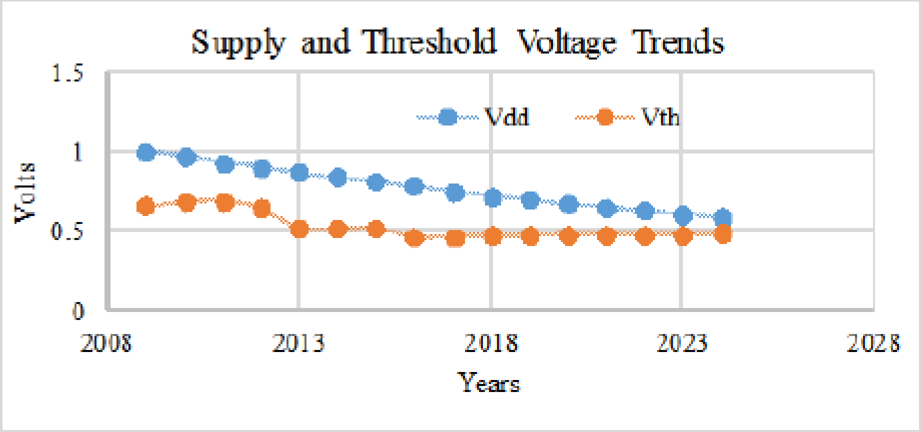
1) Problem Statement
2) Software-implemented Solutions
3) Software-implemented Testing
4) People

# Technological advances...

## Transistor Size



## Voltage Trend



Shrinking Transistor sizes and Lower operating voltages have created more powerful and energy efficient processors (devices)

IEEE Student Branch
KU Leuven Campus Brugge

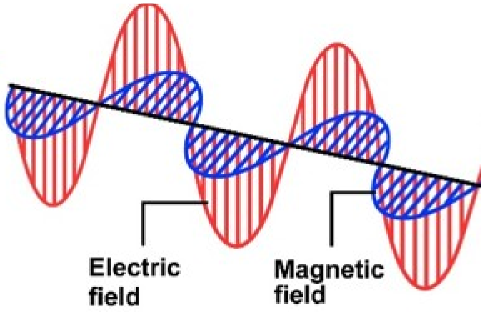
m·group

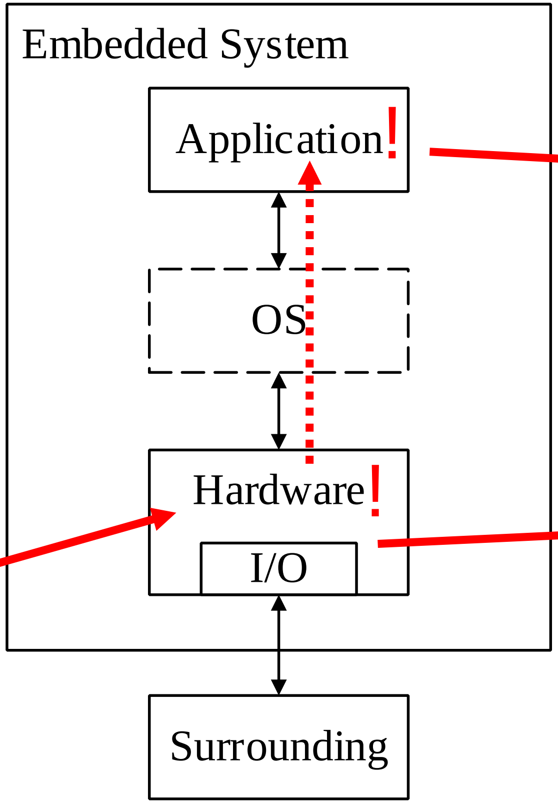
https://semiengineering.com/scaling-up-and-down/
https://www.researchgate.net/figure/Trends-for-transistor-supply-and-threshold-voltage-scaling-with-advancement-in-CMOS_fig1_274254911

# Embedded systems are vulnerable to external disturbances!

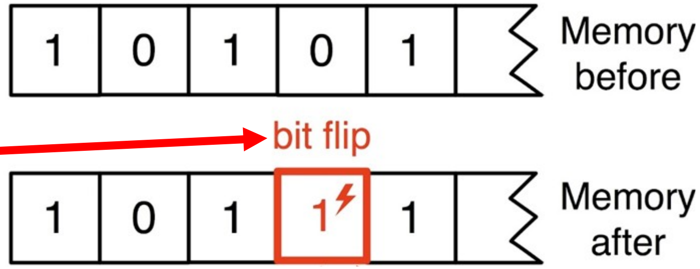
External disturbances:

- Electromagnetic Interference;
- Alpha particles, muons, etc.;
- Temperature fluctuations;
- Supply voltage spikes;
- Etc.





The introduced bit-flips can affect the application!



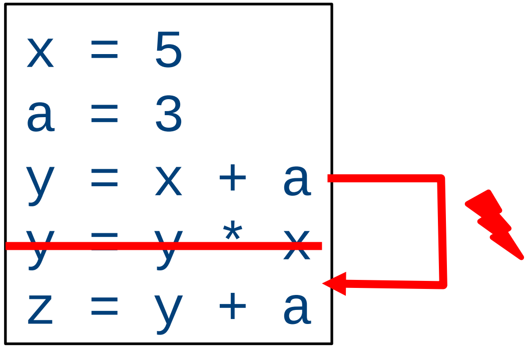
$\alpha_{\text{occurrence}} > 1/\text{day}$

G. Hubert, L. Artola, “Experimental evidence of ground albedo neutron impact on soft error rate for nanoscale devices,” *IEEE Trans. Nuclear Science*, 2018.

# Erroneous bit-flips can affect the application
# A closer look at the introduced errors

### Control Flow Error

= disruption of the execution order of instructions (unwanted jumps)



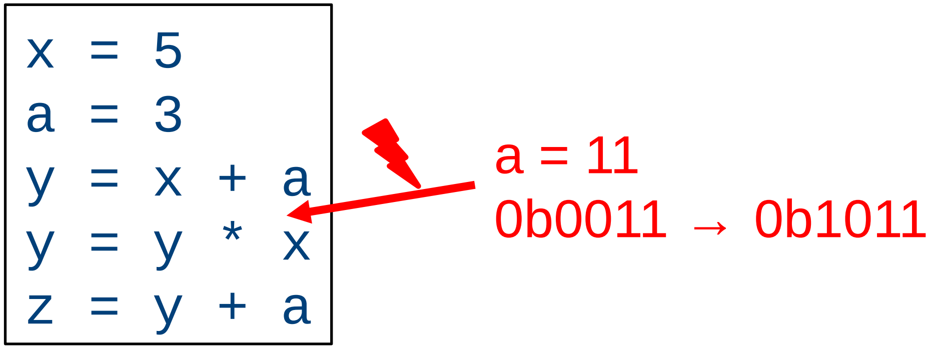
```
x = 5
a = 3
y = x + a
~~y = y * x~~
z = y + a
```

No corruption, z will be 43

Erroneous jump z will be 11, not 43

### Data Flow Error

= corruption of input, intermediate or output values



```
x = 5
a = 3
y = x + a
y = y * x
z = y + a
```

a = 11
0b0011 → 0b1011

No corruption, z will be 43

a corrupted to 11, z will be 51, not 43

# Erroneous bit-flip detection

**Hardware Detection**
= duplicate systems and add voting logic

+ Effective

- High cost

- Low flexibility

**Software-implemented Detection**
= insert extra instruction that enable error detection

+ High flexibility

+ No need for extra hardware
  = lower cost / system

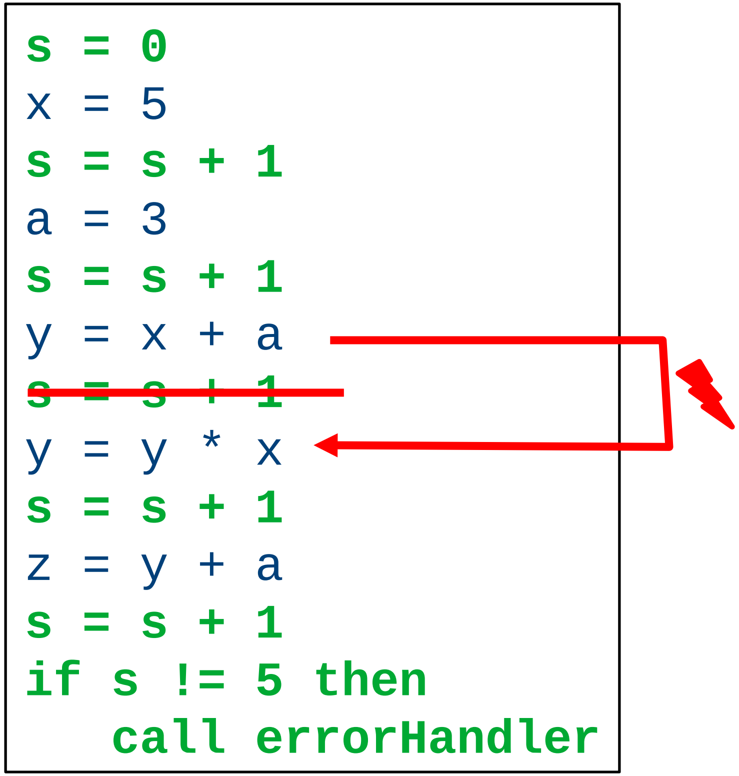
- Difficult to implement

- Introduce overhead

# Overview

1) Problem Statement
2) Software-implemented Solutions
3) Software-implemented Testing
4) People

# Control Flow Error Detection



```
s = 0
x = 5
s = s + 1
a = 3
s = s + 1
y = x + a
~~s = s + 1~~
y = y * x
s = s + 1
z = y + a
s = s + 1
if s != 5 then
   call errorHandler
```

No corruption,
z will be 43

Erroneous jump
s will be 4, not 5
→ error handler is called and
the system can deal with the error

# Data Flow Error Detection

```
x = 5
x_dup = 5
a = 3
a_dup = 3
y = x + a
y_dup = x_dup + a_dup
y = y * x
y_dup = y_dup * x_dup
z = y + a
z_dup = y_dup + a_dup
If z != z_dup then
   call errorHandler
```

No corruption,
z will be 43

a = 11
0b0011 → 0b1011

a corrupted to 11,
z will be 51, z_dup will be 43
→ error handler is called and
the system can deal with the error

# Research

- Control Flow Error Detection (Jens Vkb)
  - New and better Techniques: RASM and RACFED
  - Compiler plugin to automate technique implementation
    *(not discussed today)*
- Data Flow Error Detection (Venu B. Thati)
  - New and better Techniques: FDSC and ILDCC

# Overview

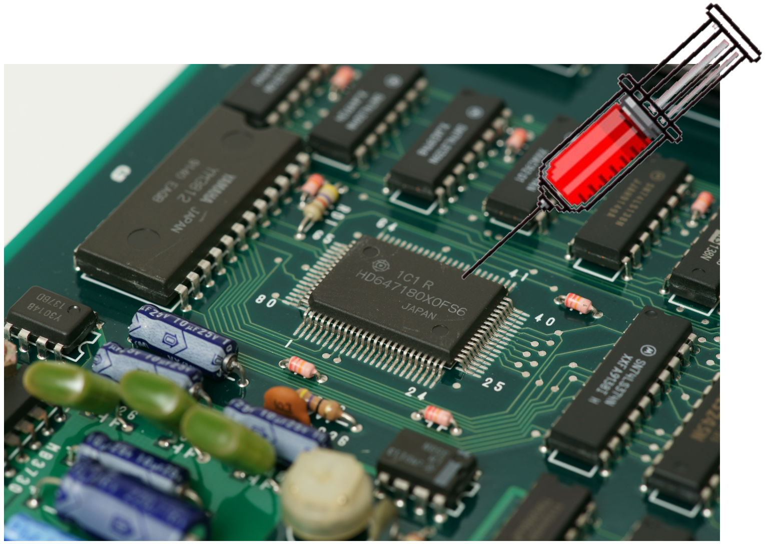
1) Problem Statement
2) Software-implemented Solutions
3) Software-implemented Testing
4) People

# Detection technique verification and validation

- Once implemented, the detection mechanism must be validated → faults must occur in the system

Option 1: ad hoc

Option 2: fault injection



# Research

- Developed new fault injection processes
- Implemented a fault injection tool
  - For physical targets (Python) connecting to on-chip debugger
  - For simulated targets (C++) from the Imperas simulator

# Overview

1) Problem Statement
2) Software-implemented Solutions
3) Software-implemented Testing
4) People

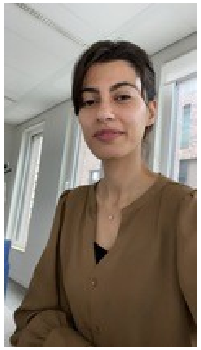
# Overview of Researchers

prof. Jeroen Boydens
prof. Hans Hallez

dr. ing. Jens Vankeirsbilck

dr. ing. Venu Babu Thati
(graduated Apr. 2020)

ing. Brent De Blaere



Mohaddaseh Nikseresht

IEEE Student Branch
KU Leuven Campus Brugge

m·group