CIRCA:
An Approximate Computing Tool Flow

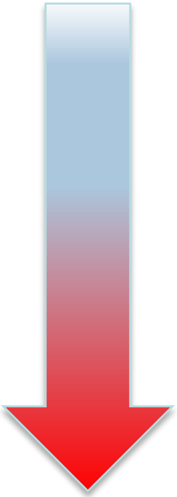
Project Group for CS & CE
Starting WS'20/21

Linus Witschen
witschen@mail.upb.de
Computer Engineering Group

What Is Approximate Computing?

How quickly can you calculate?

- ○ Is 47.2 divided by 1.3 greater than 1?
- ○ Is 47.2 divided by 1.3 greater than 35?
- ○ What is 47.2 divided by 1.3?



increasing accuracy (effort)

⇨ Required accuracy depends on the problem,
yet computers use always the same accuracy (effort)!

Can Applications Tolerate Lower Accuracy?

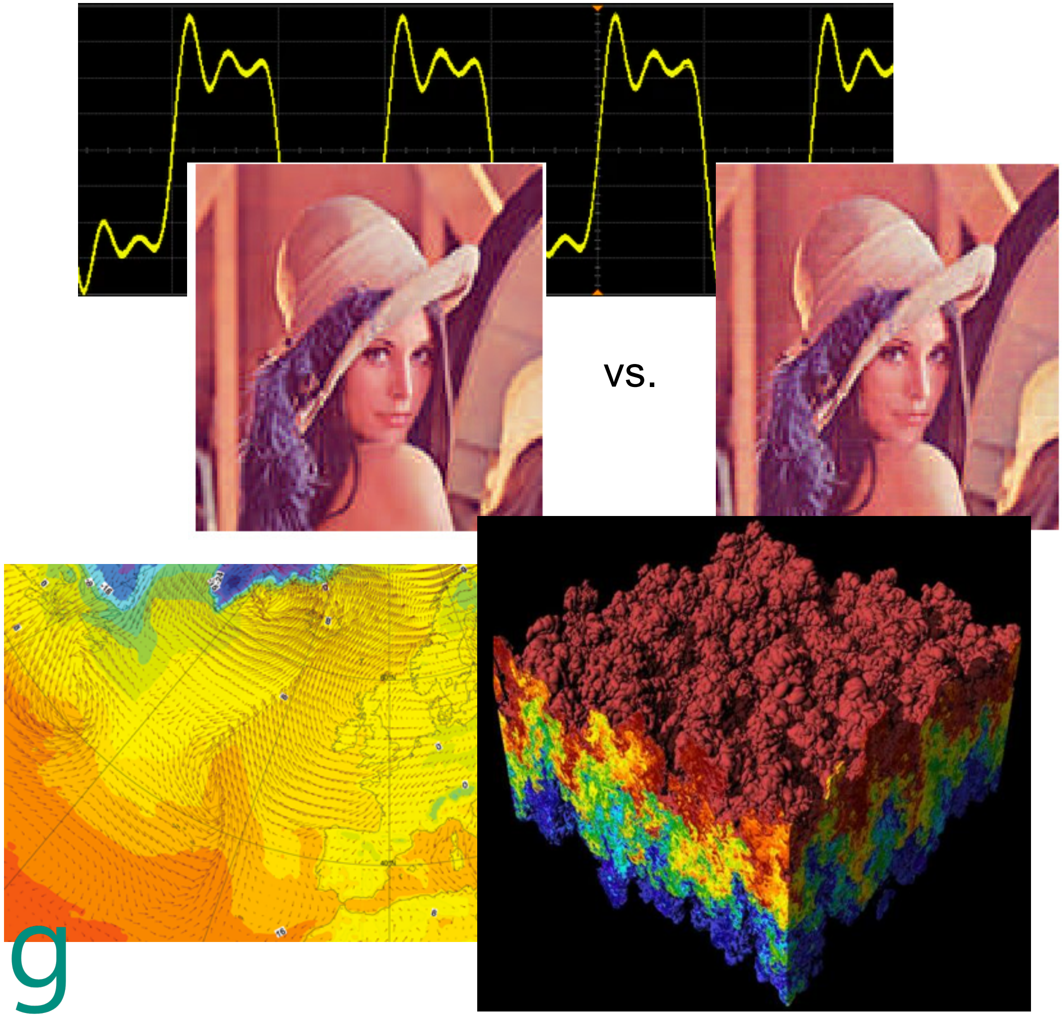
YES, definitely, when …

… input data is noisy, imprecise, or incomplete

… human perception is the limit

… exact or optimal results are out of reach anyway

… the correct answer is not known



Google

Bing

Why Do We Need More Efficient Computing?

Dennard Scaling

The power dissipation per unit area remains constant.

Moore's Law

The number of transistors per unit area doubles every 18-24 months.

…. stopped around 2003/05 !

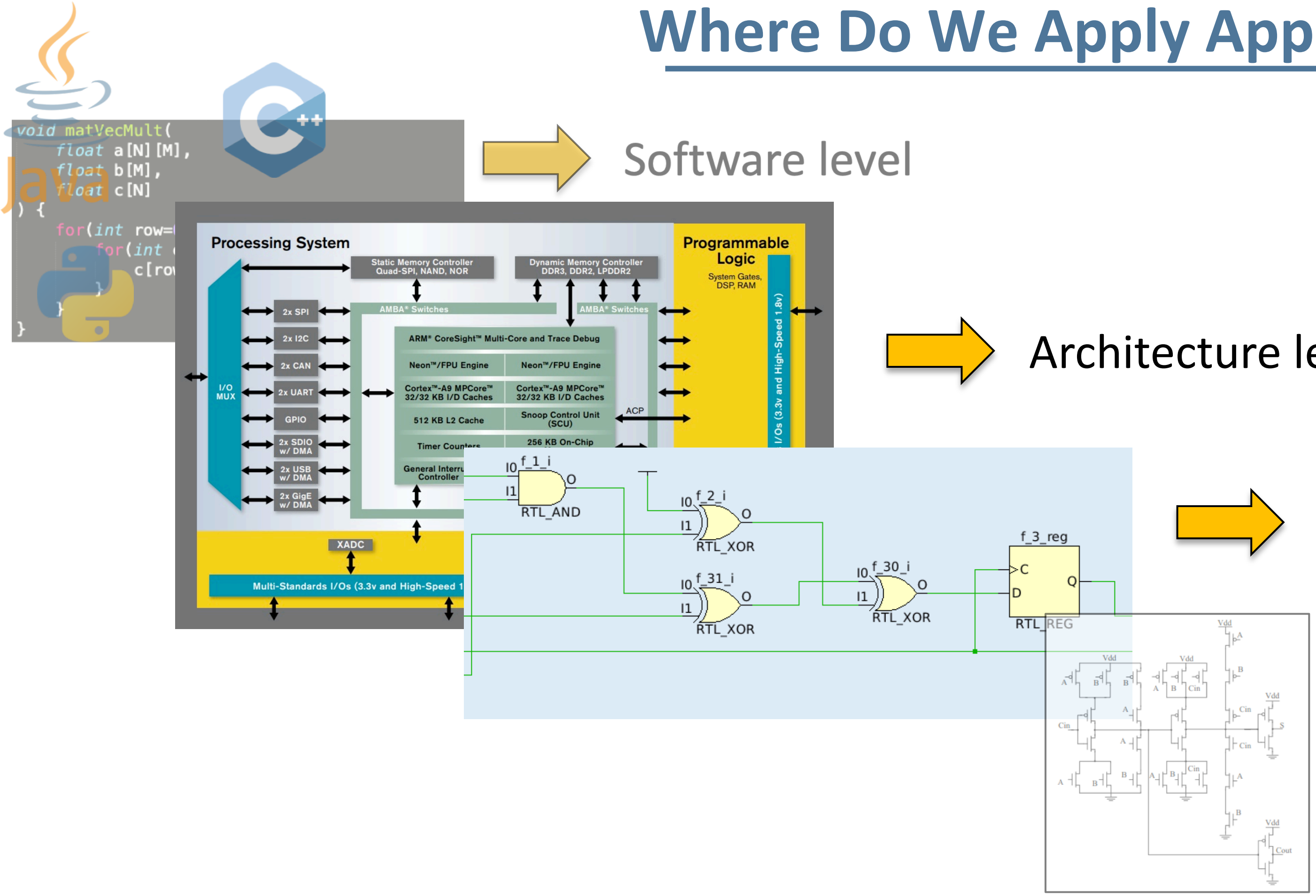
... slowing down since some years !

Progress in process technology comes to an end!

New ways needed to improve performance under strict power/energy constraints!

Approximate Computing can be used to trade off accuracy for lower power / energy OR higher performance OR smaller chip size

Where Do We Apply Approximations?



```
void matVecMult(
    float a[N][M],
    float b[M],
    float c[N]
) {
    for(int row=
        for(int 
            c[ro
        }
    }
}
```

Software level

Architecture level

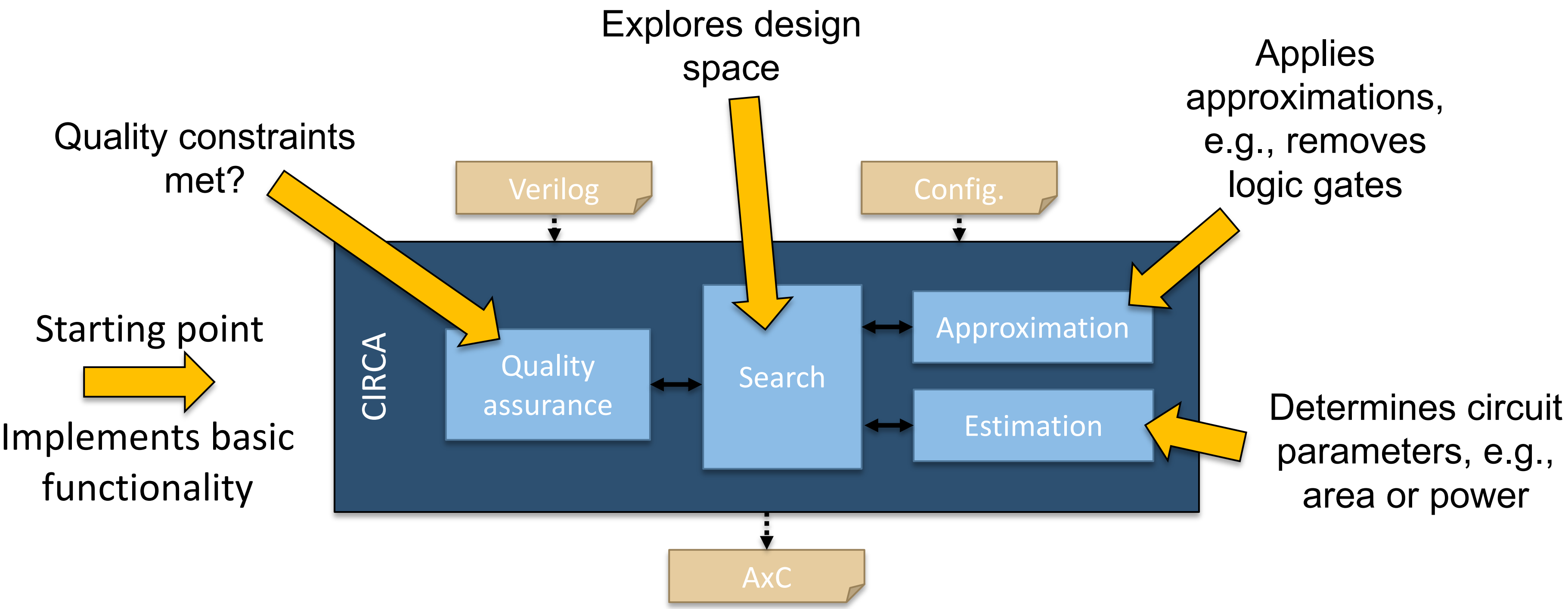
Logic level

Circuit level

We want to approximate the architecture and the logic of a hardware system!

How can we automatize the process of approximate circuit generation?

Development of an Approximate Computing Tool Flow



Quality constraints met?

Explores design space

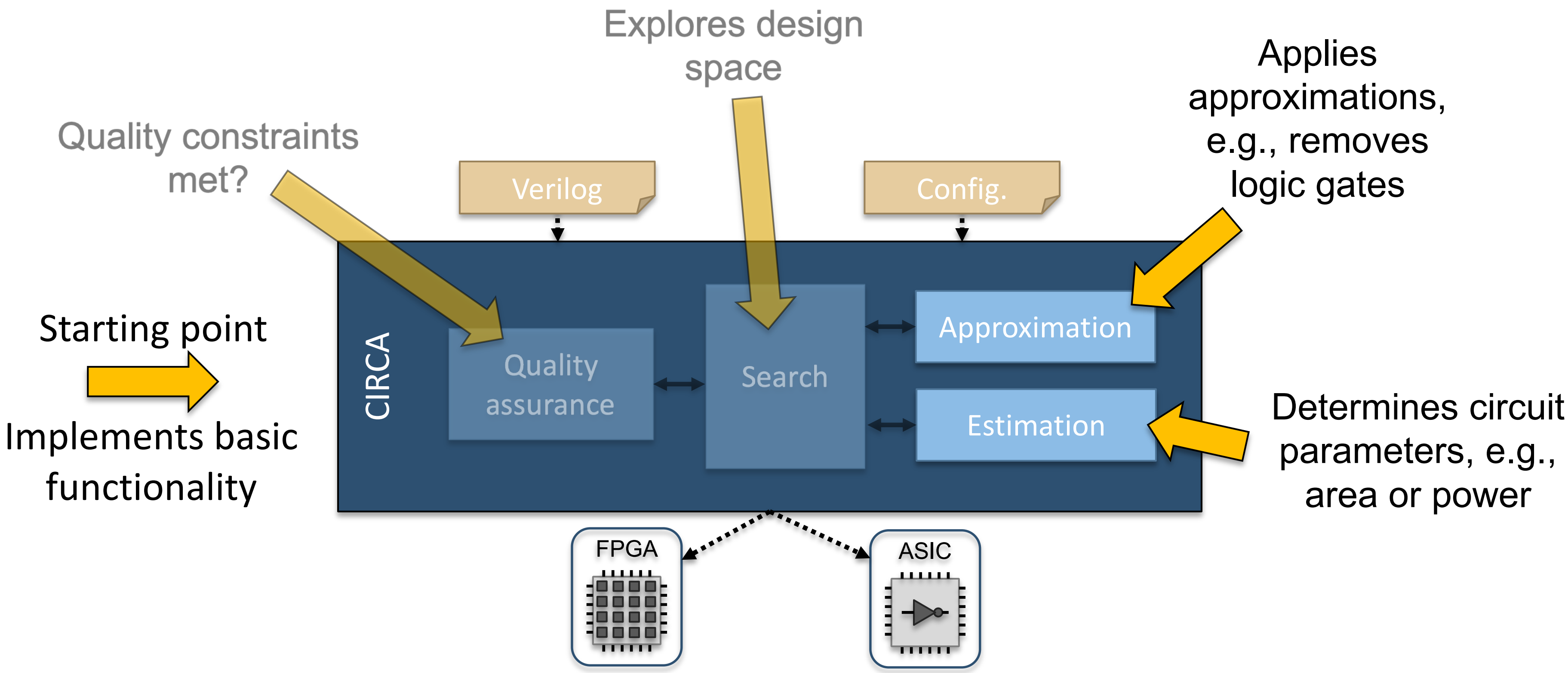
Applies approximations, e.g., removes logic gates

Starting point

Implements basic functionality

Determines circuit parameters, e.g., area or power

Development of an Approximate Computing Tool Flow



First semester: extend CIRCA

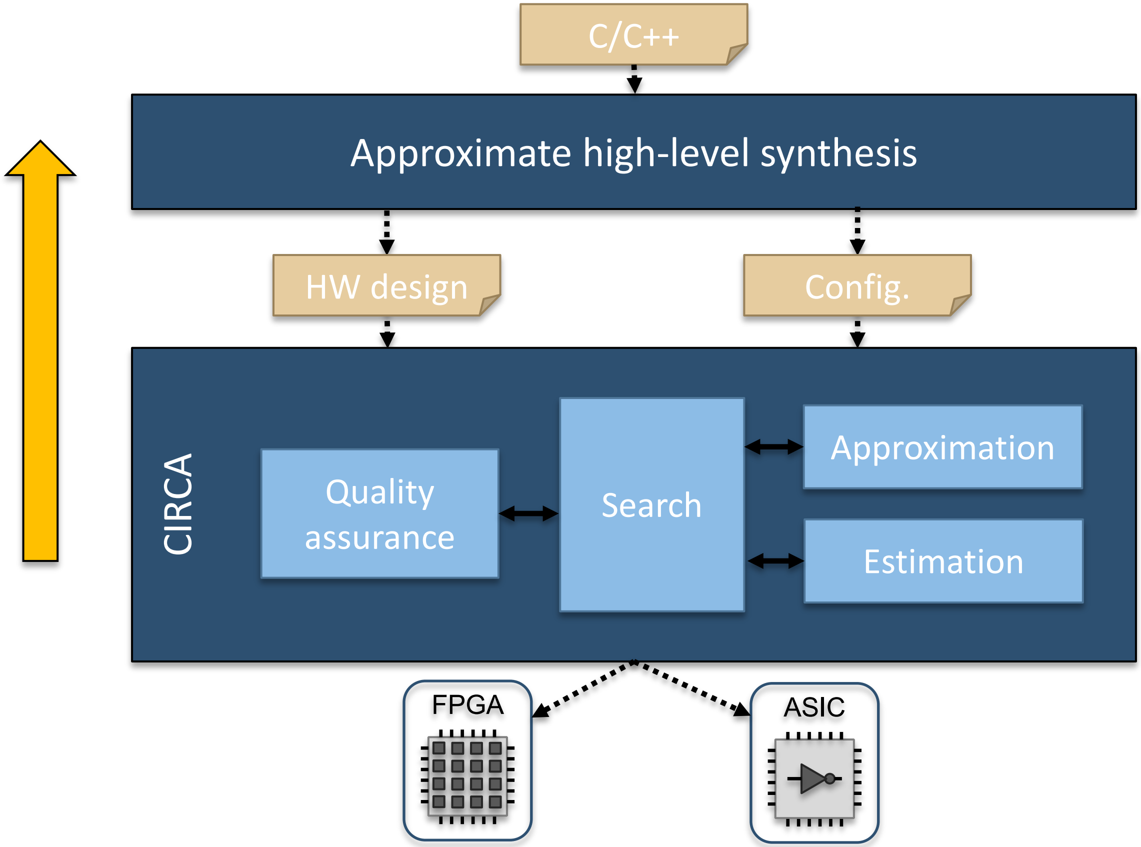
Focus: approximation methods, target metrics, & integration of back-end tools

Development of an Approximate Computing Tool Flow

Raise level of abstraction

Simpler to describe complex systems

Enables approximations on system level



Second semester: integration of AHLS into the AxC flow

Focus: AxC flow, design partitioning , & concepts for different abstraction levels

Two Subgroups, Two Focuses

Job descriptions

AxC tool flow (SW-oriented)

- Design & implementation of the tool flow

AxC methods (approximate HW-oriented)

- Development & implementation of AxC methods

No prerequisites, but a plus if experienced in

- Python, C/C++, & Verilog
- The (high-level) synthesis process
- (Approximate) hardware design
- Development of complex software

Curriculum Vitae

Programming languages

- Python
- C/C++
- Verilog

Tools

- Open-source tools
  - Berkeley's ABC, Yosys, & HLS tools
- Commercial tools
  - Synopsys Design Compiler & Xilinx Vivado

Skills & experiences

- Expert in Approximate Computing & (high-level) synthesis
- Project management, organization, & presentation
- Running experiments on a compute cluster

AxC tool flow

- Developing large software project for hardware domain
- Modelling & compilation of a complex tool flow

AxC methods

- Developing concepts for efficient hardware systems
- Extending synthesis tools by custom algorithms

Contact Information & Further Material

- Visit our project group website for more information
- Attend the **Approximate Computing lecture** in the **winter semester'20/21**!
- Visit the CIRCA website
- Contact us

Linus Witschen
witschen@mail.upb.de
Computer Engineering Group

Prof. Marco Platzner
platzner@upb.de
Head of the Computer Engineering Group