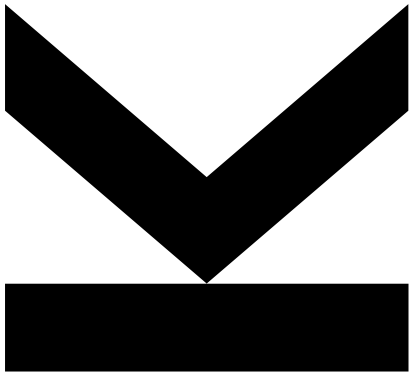
Efficient Post-Silicon Run-Time Error Detection for Systems-on-Chip



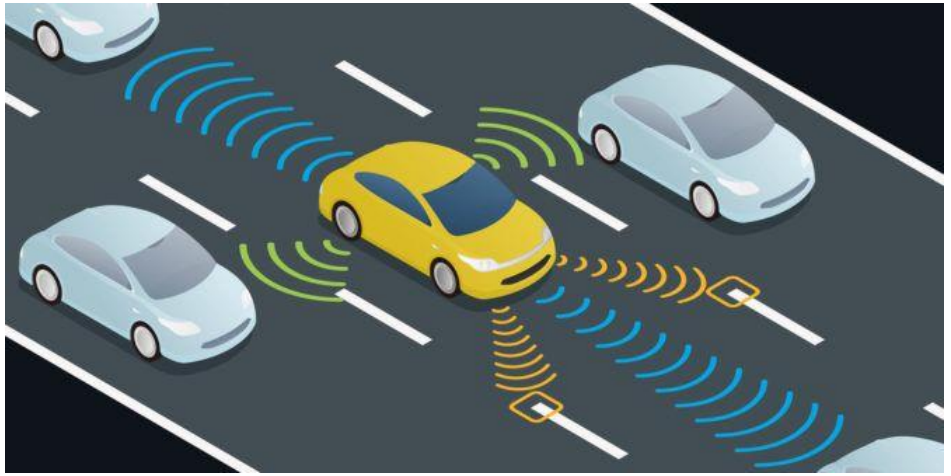
Sebastian Pointner, Martin Brunner, Rainer Findenig & Robert Wille
Johannes Kepler University Linz & Infineon Technologies Linz

TuZ Workshop
22.02.2021, Virtual Conference

JOHANNES KEPLER UNIVERSITY LINZ
Altenberger Straße 69
4040 Linz, Austria
jku.at

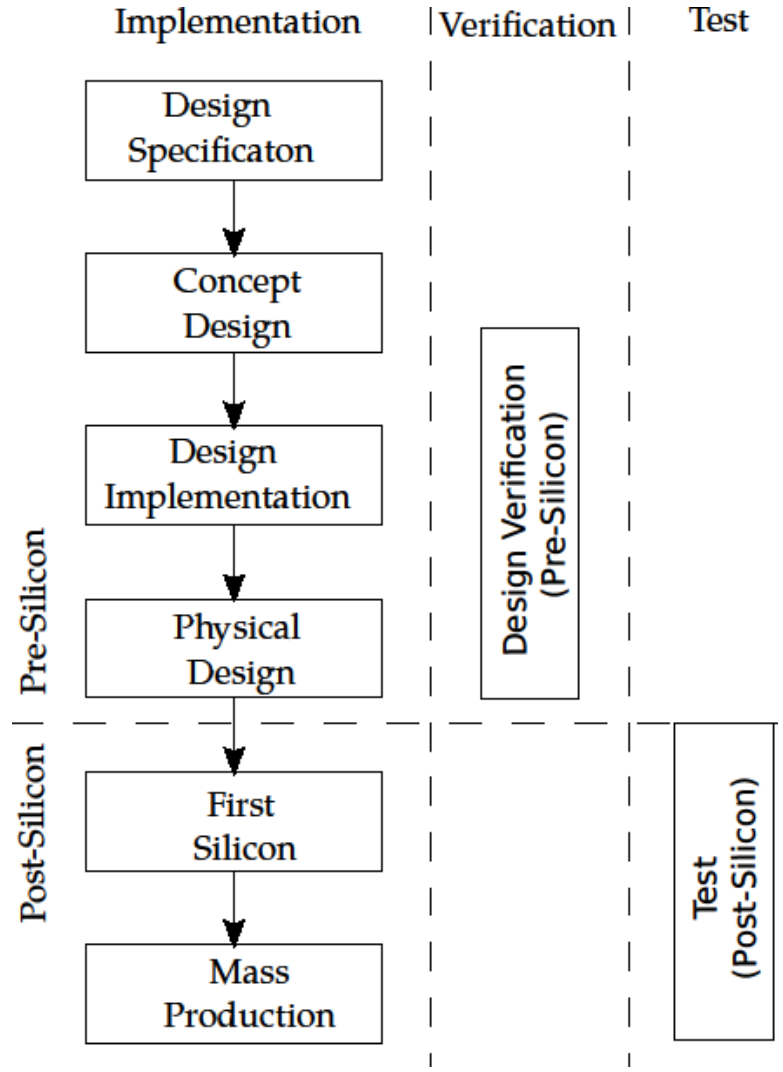
Big Picture of this work

- Ensure that a system works as intended
- Systems which are safe (e.g., functional safety)
- Systems which are failsafe (e.g., redundant systems)
- Systems capable to react to certain scenarios



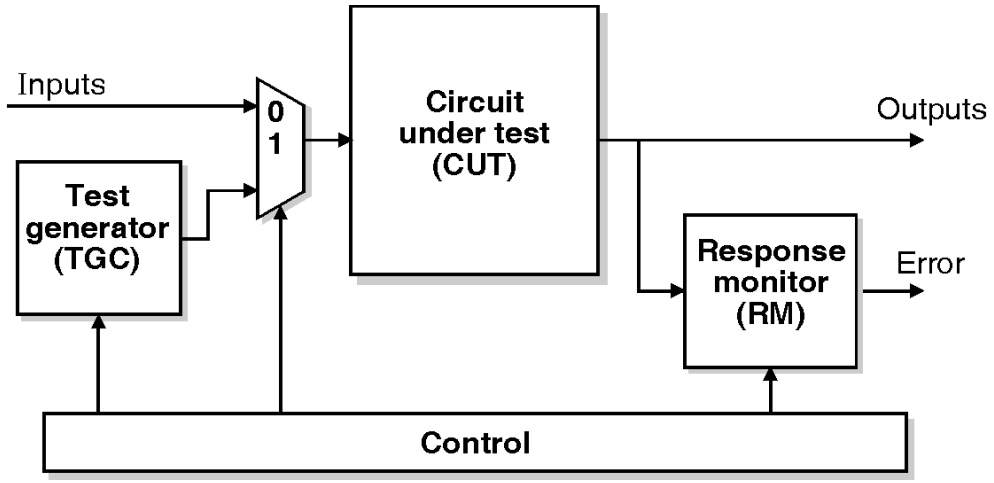
Ensuring Safe Systems: Design Phase

- Classical approach
  - I.e., ISO26262 flow
- Pre-Silicon Verification:
  - Do we design the thing right?
- Post-Silicon Test:
  - Do we have the right thing?



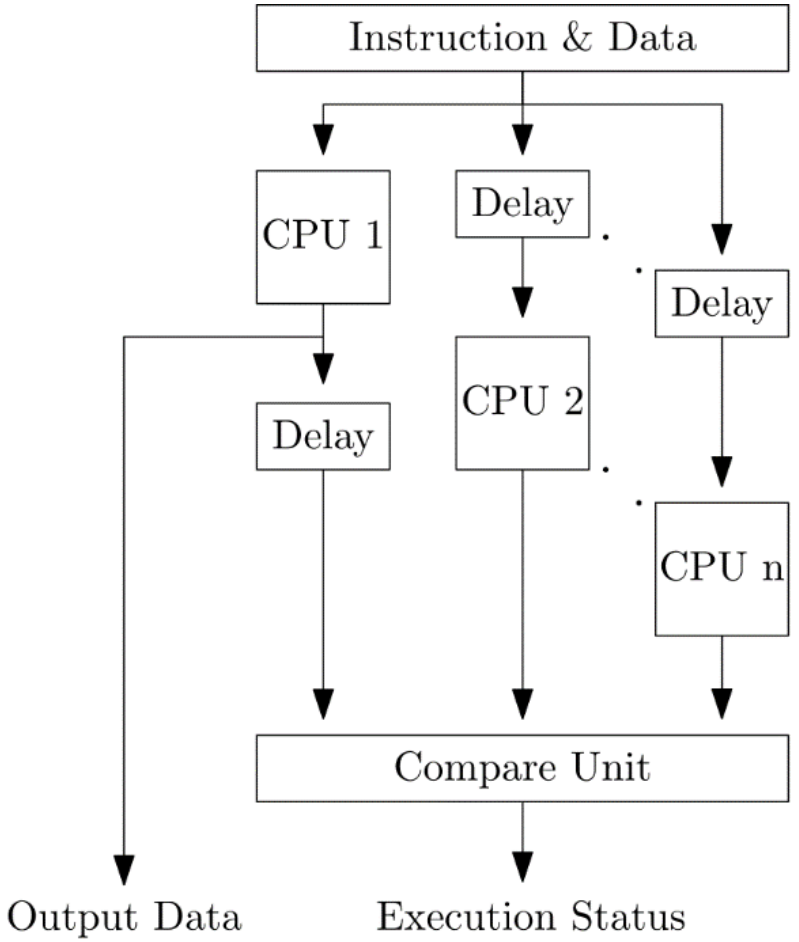
Ensuring Safe Systems: After Deployment

- Post Silicon Test
  - Covers only a point of time
- Utilizing BIST capabilities
  - Check certain properties during e.g. boot routine
  - Does also not cover entire run-time
- Goal: run-time error detection ☺



Ensuring Safe Systems: Using Redundancy

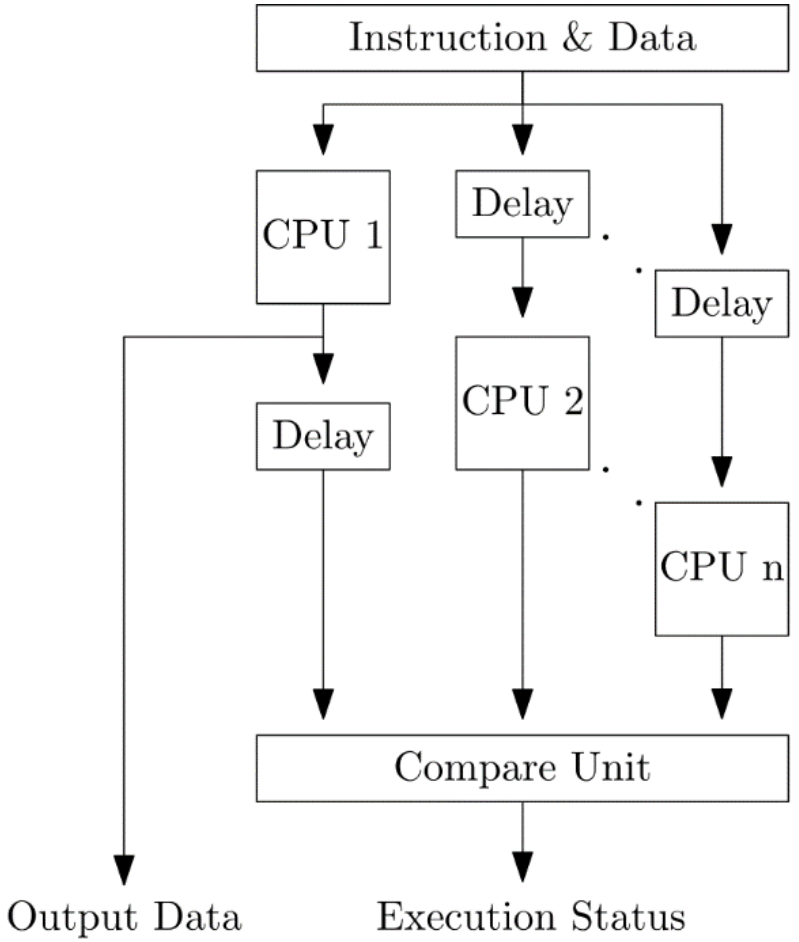
- Lock-step approach
  - Utilize more instances e.g. of a CPU
  - Compare result for error detection
  - Insert delay to exclude external influences



Ensuring Safe Systems: Using Redundancy

- Lock-step approach
  - Utilize more instances e.g. of a CPU
  - Compare result for error detection
  - Insert delay to exclude external influences

- Advantages:
  - Simple and robust design ☺

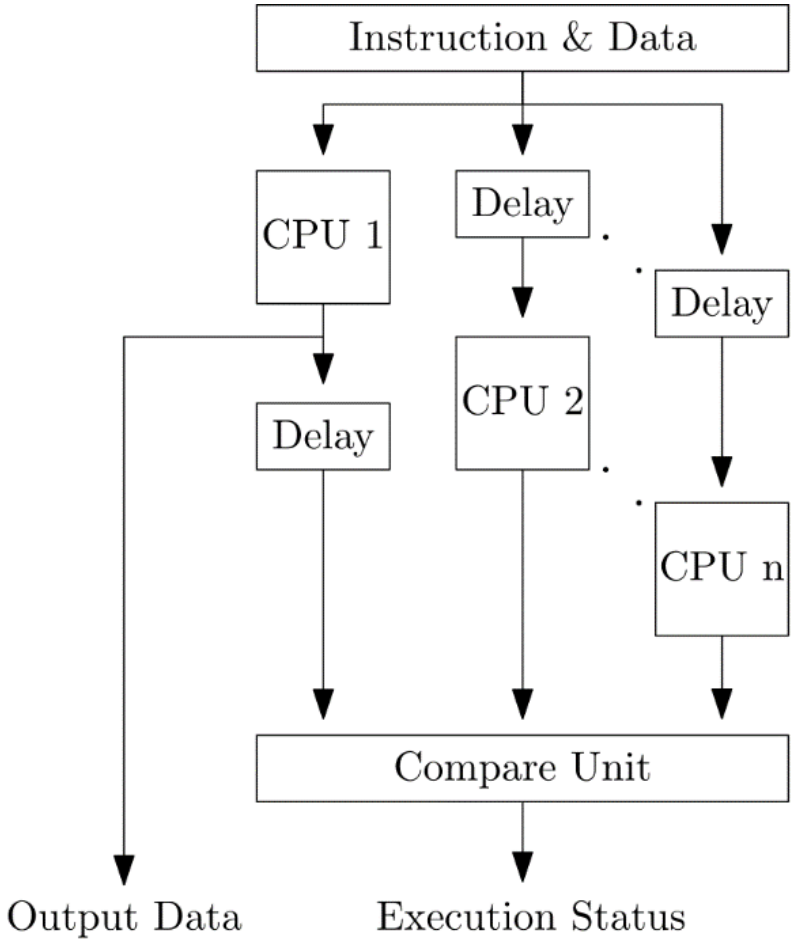


Ensuring Safe Systems: Using Redundancy

- Lock-step approach
  - Utilize more instances e.g. of a CPU
  - Compare result for error detection
  - Insert delay to exclude external influences

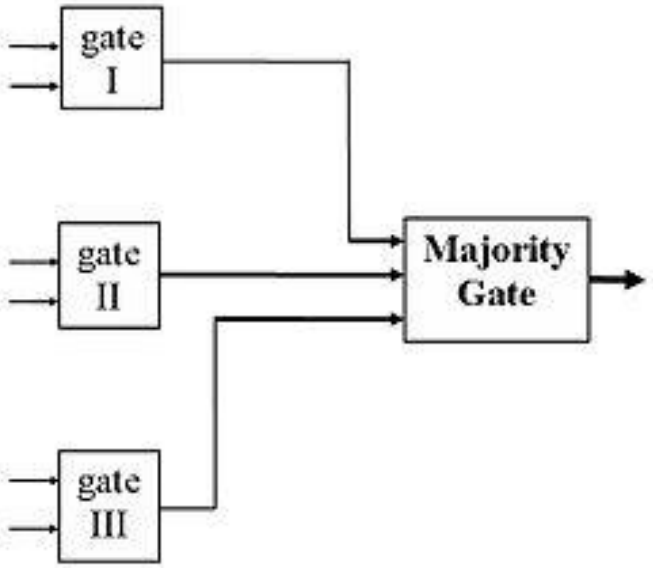
- Advantages:
  - Simple and robust design ☺

- Disadvantages:
  - More chip area is needed ☹
  - Higher power consumption ☹
  - Delayed computation ☹



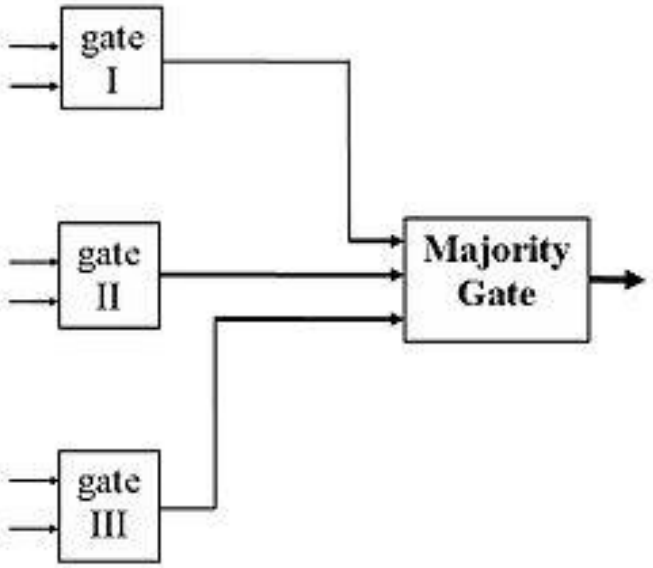
Analysis of the Lock-Step Approach I

- Redundant usage of CPUs
- Same inputs for the CPUs
- Same firmware used for the CPUs
- Delay to exclude external influences



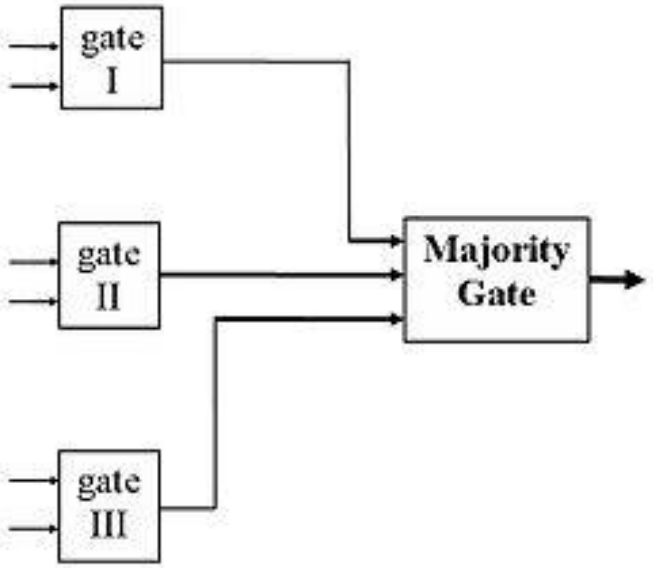
Analysis of the Lock-Step Approach I

- Redundant usage of CPUs
- Same inputs for the CPUs
- Same firmware used for the CPUs
- Delay to exclude external influences
- Idea: Make usage of firmware to get current execution information



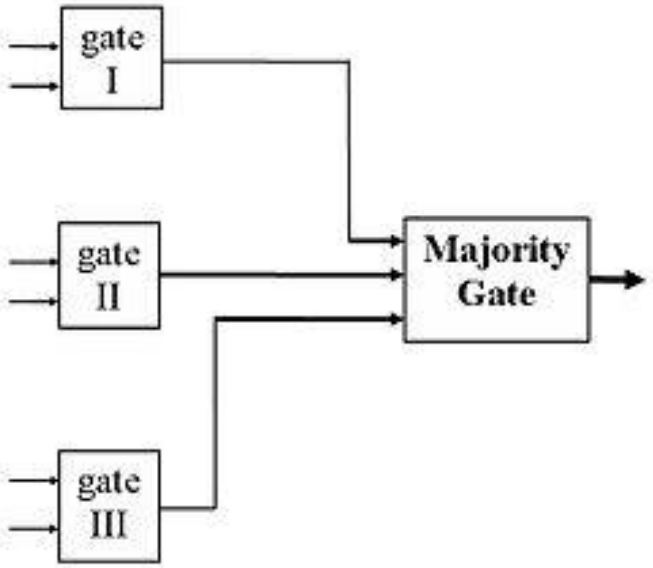
Analysis of the Lock-Step Approach II

- Both CPUs run the same firmware
- Both CPUs have seen the same history of inputs
- Both CPUs should be in the same execution state



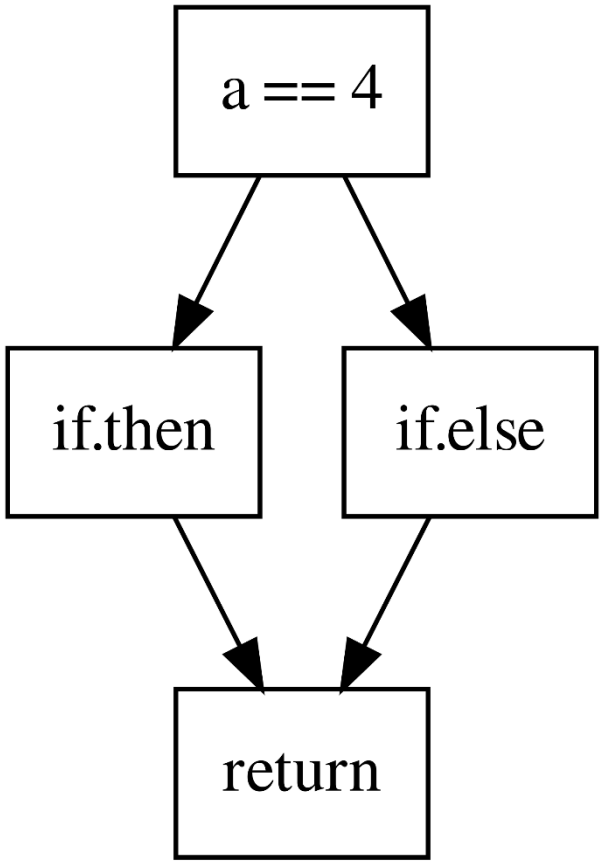
Analysis of the Lock-Step Approach II

- Both CPUs run the same firmware
- Both CPUs have seen the same history of inputs
- Both CPUs should be in the same execution state
- Idea: Explore the firmware for valid states



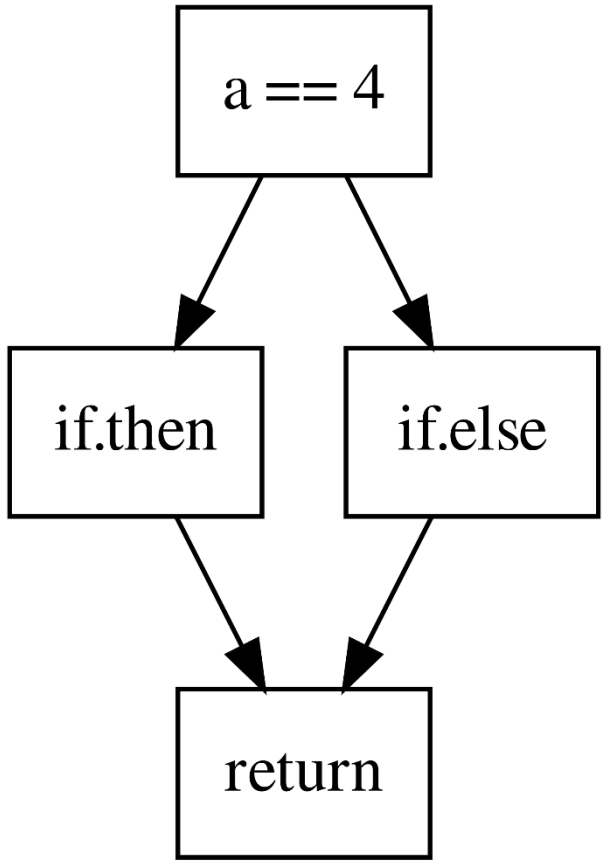
Symbolic Execution

- Generate Control Flow Graph symbolically
- Explore firmware’s state space
- Utilize symbolic values as placeholders
- Decision finding based on reasoning engines



Symbolic Execution

- Generate Control Flow Graph symbolically
- Explore firmware's state space
- Utilize symbolic values as placeholders
- Decision finding based on reasoning engines

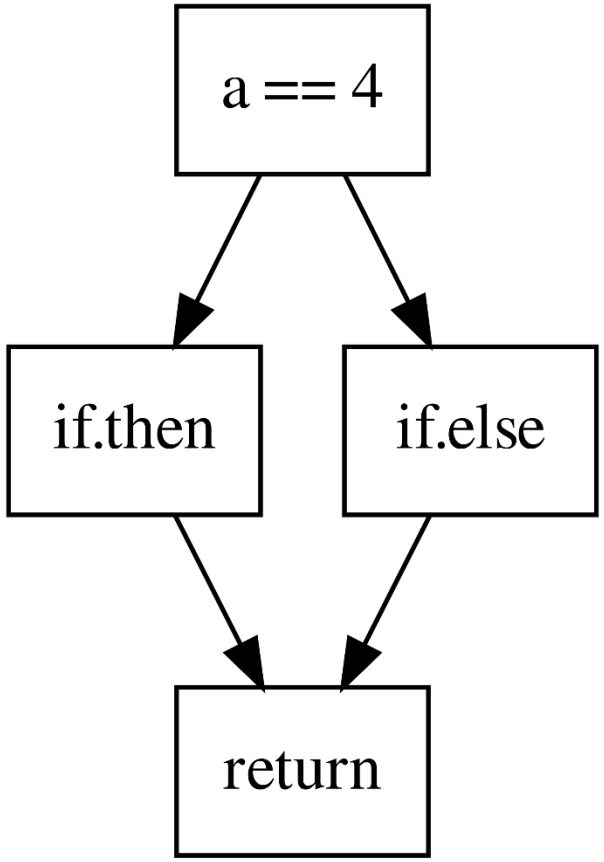



```
(declare-fun mem_1 () (_ BitVec 32))
(assert (= mem_1 #x00000004)
```

```
(declare-fun mem_1 () (_ BitVec 32))
(assert (= (= mem_1 #x00000004) false))
```

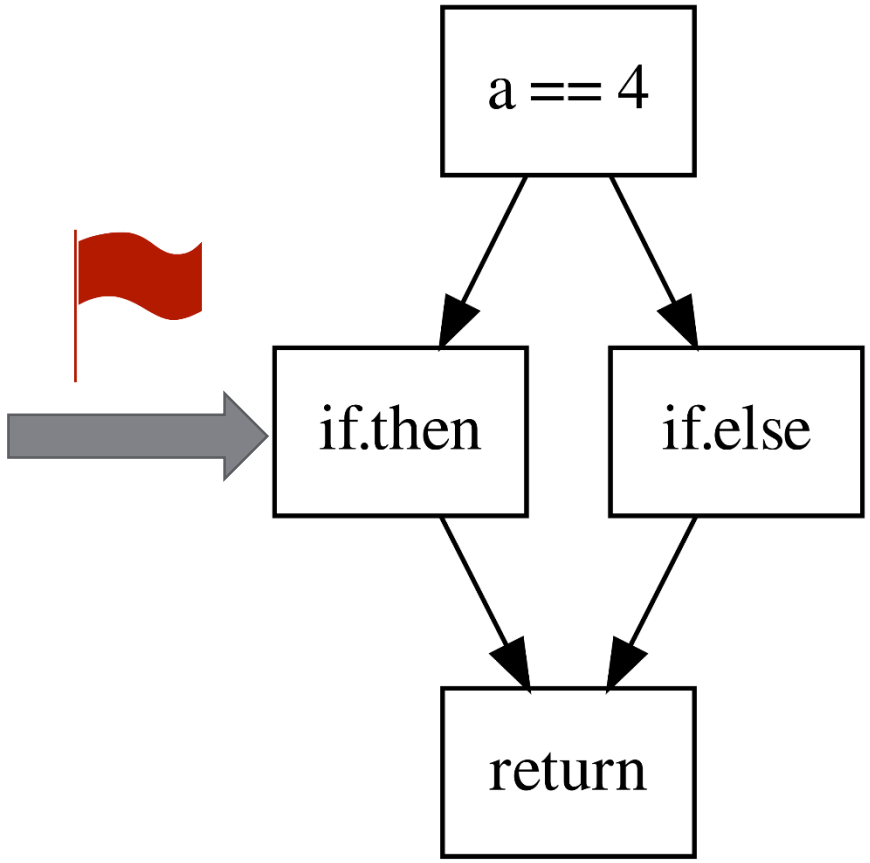
General Idea: Utilize State Space

- Utilize the explored states
- Is the state the system has reached valid?



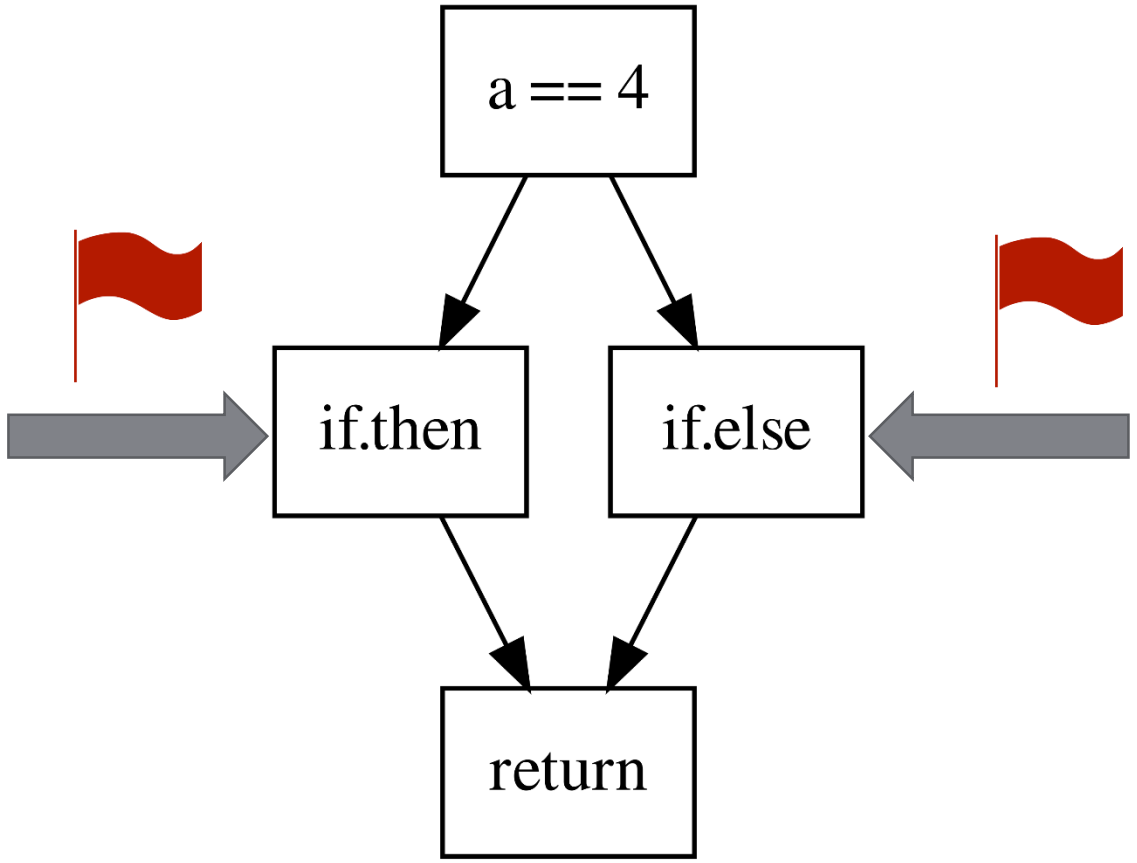
General Idea: Utilize State Space

- Utilize the explored states
- Is the state the system has reached valid?
- Taken branch as state



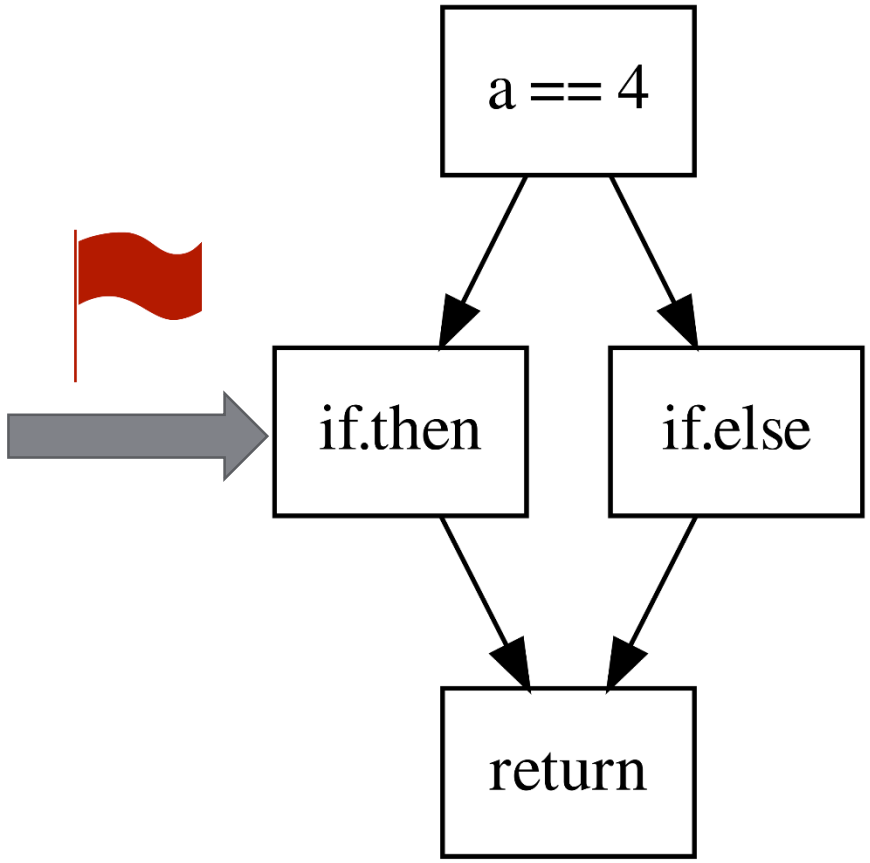
General Idea: Utilize State Space

- Utilize the explored states
- Is the state the system has reached valid?
- Taken branch as state
- Non-taken branch



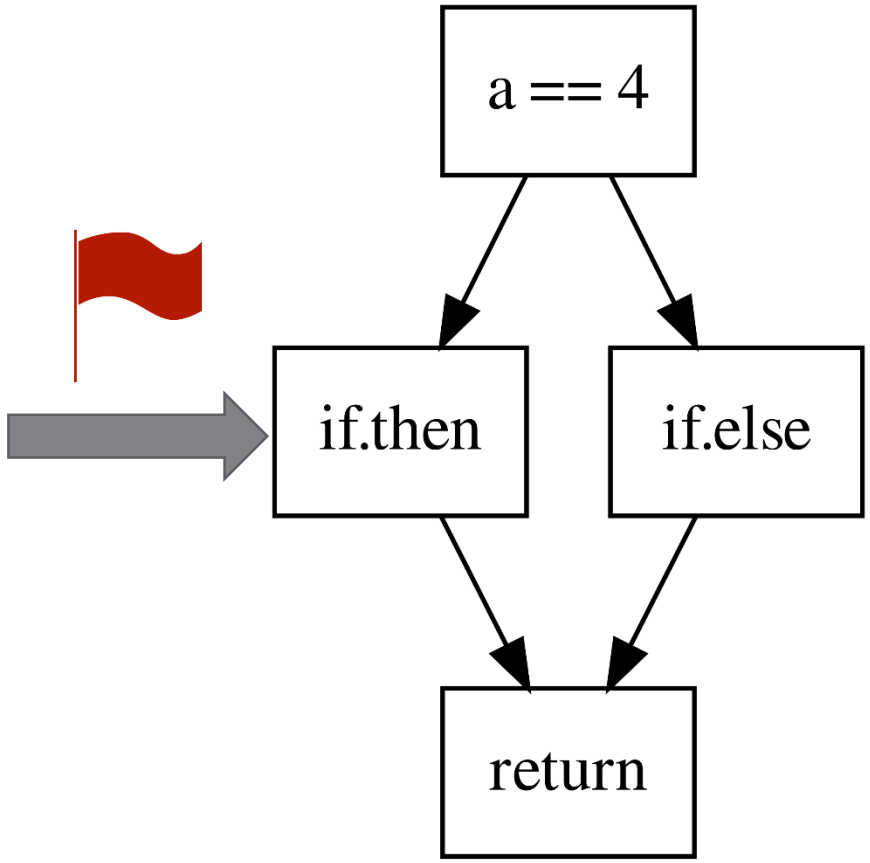
General Idea: Utilize State Space

- Utilize the explored states
- Is the system allowed to take the branch?
- Considering the execution history?



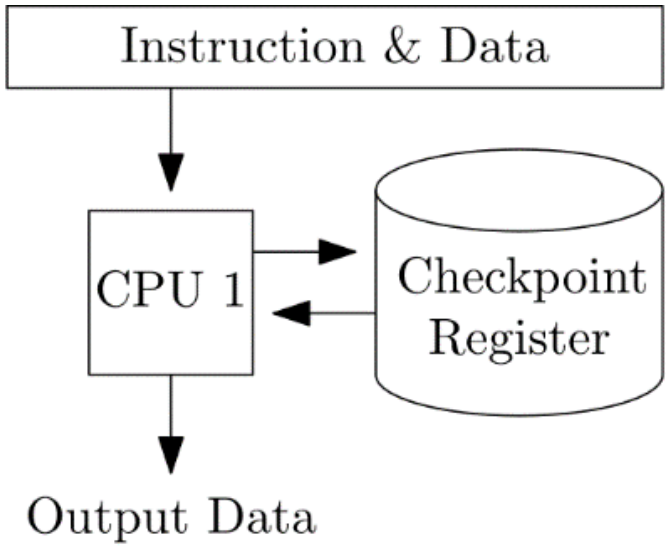
General Idea: Utilize State Space

- Utilize the explored states
- Is the system allowed to take the branch?
- Considering the execution history?
- Idea:
  - Pre-compute valid execution states
  - Compare current state with stored values



General Idea: Utilize Checkpoints

- Insert check points during symbolic execution
- Hash based on execution history
- Stored in check point registers
- System can compare:
  - Value stored in check point register
  - Run-time calculated hash value



Check Point Insertion

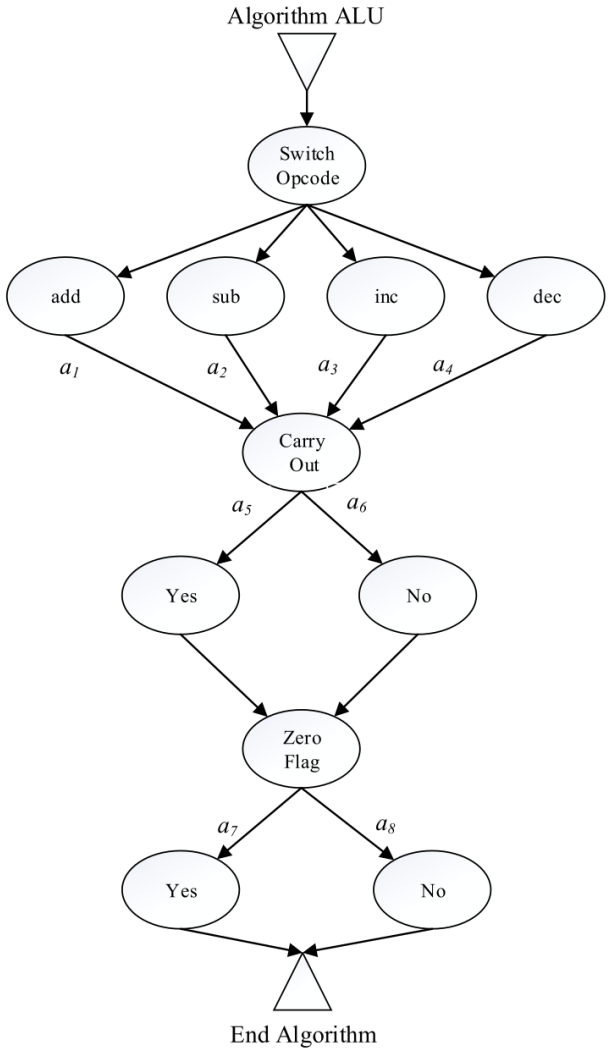
- Automatically after each branch

- Only for certain states
  - Marked by the designed
  - Using Checkpoint Functions

- Hash value for every checkpoint to be stored

Check Point Insertion: Full Coverage

- Automatically after each branch
- ALU CFG already leading to 8 register
- Does it make sense for every branch?
- Hardware overhead vs. coverage

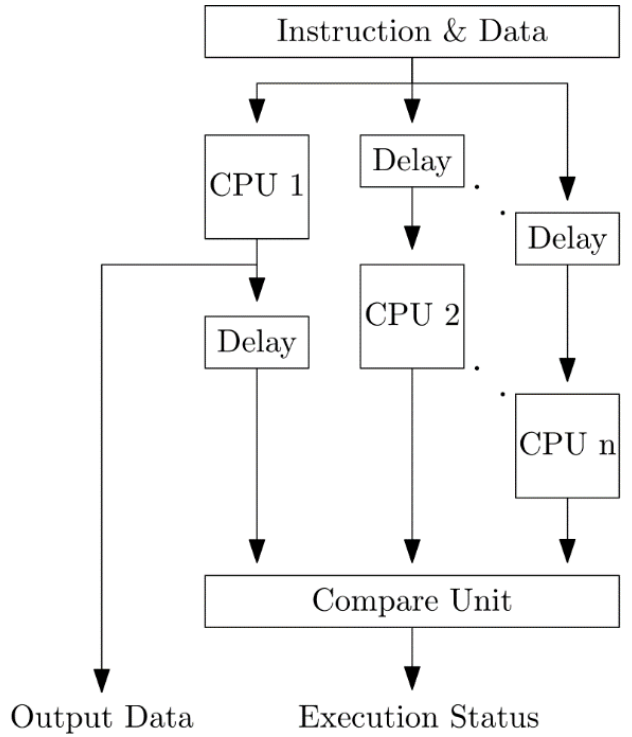


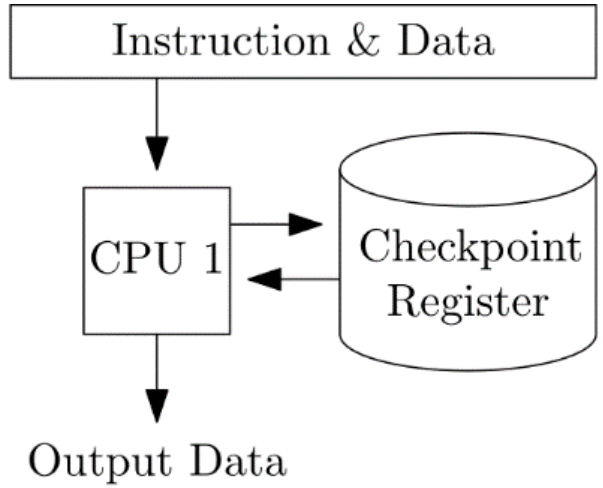
Check Point Insertion: Designer Guided

- Checkpoint after each branch needed?
- Designer know functional safety critical code sections
- Guide the symbolic execution for the insertion
- Place “Checkpoint Functions” into the firmware
- Trade-off: hardware overhead vs. coverage

Advantages/Disadvantages

- Lockstep:
  - Duplicating or tripling the entire CPU

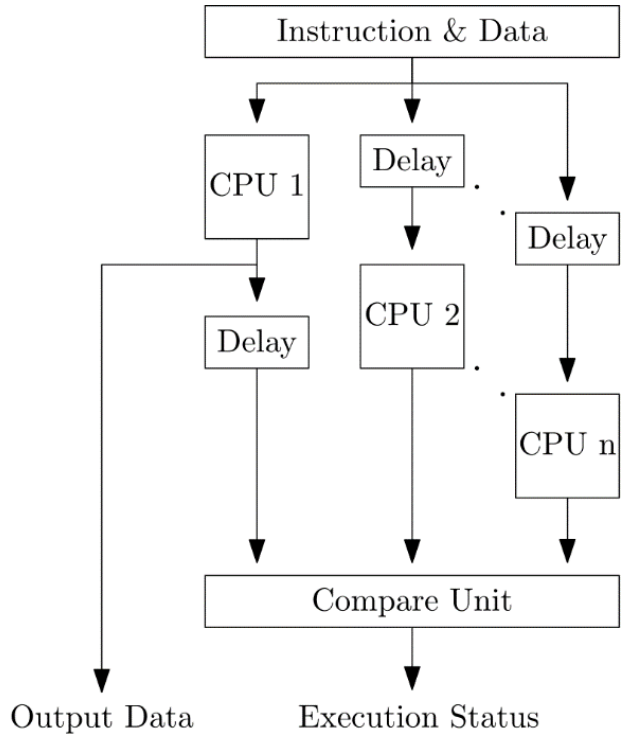


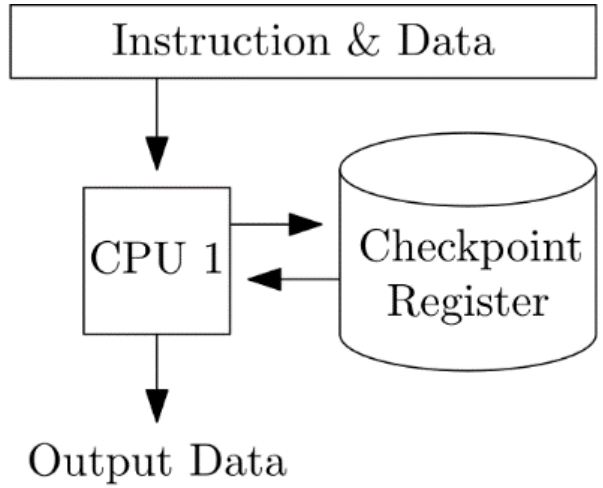


Advantages/Disadvantages

- Lockstep:
  - Duplicating or tripling the entire CPU

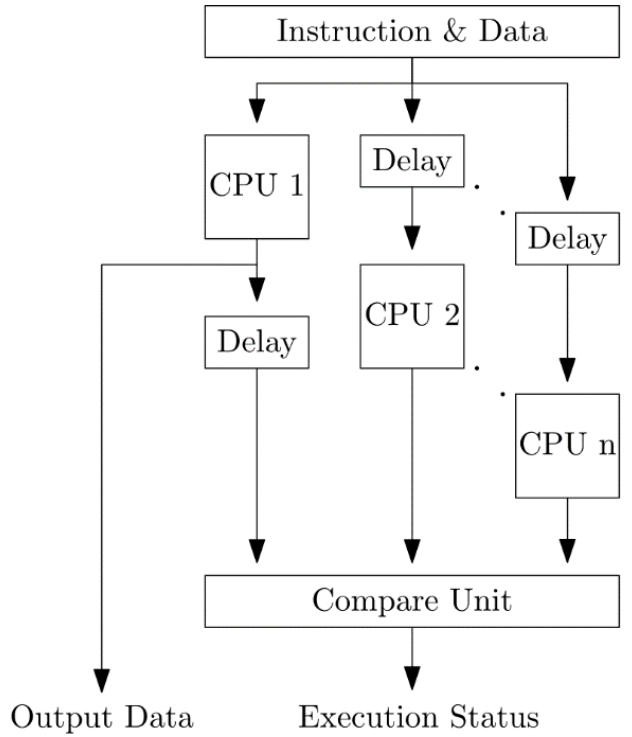
- Program Analysis approach:
  - Additional memory needed

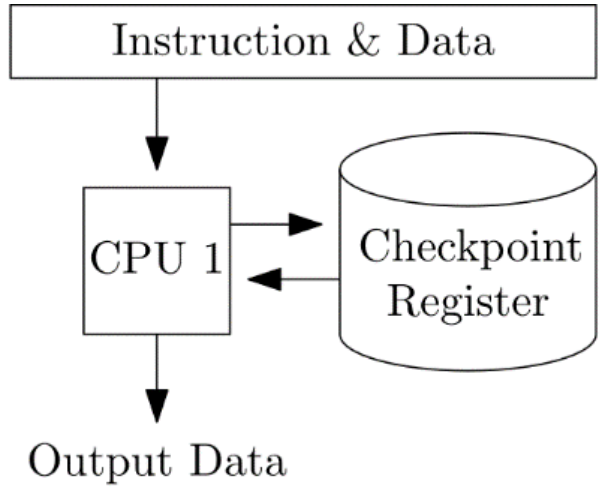




Advantages/Disadvantages

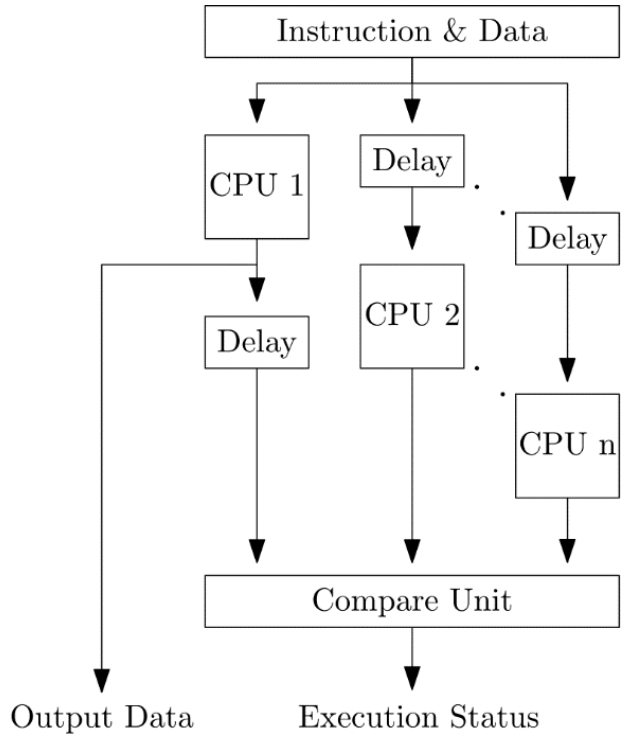
- Lockstep:
  - Duplicating or tripling the entire CPU
- Program Analysis approach:
  - Additional memory needed
- Pros of the approach:
  - Easy to realize
  - Less hardware overhead

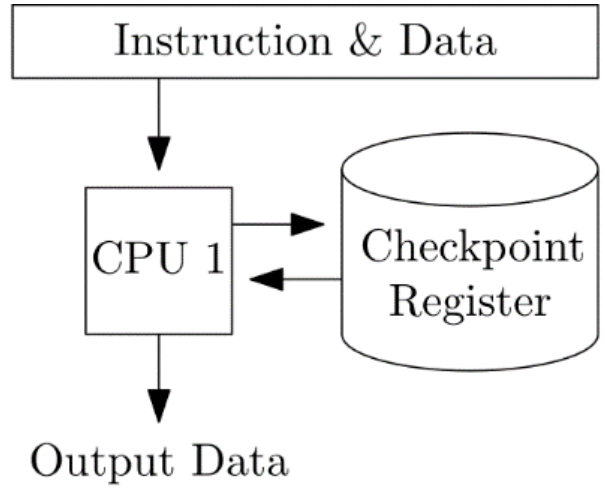




Advantages/Disadvantages

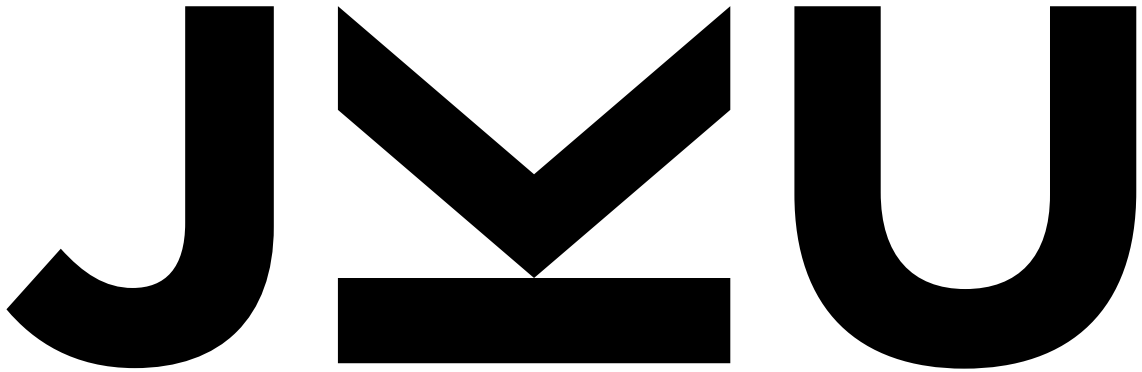
- Lockstep:
  - Duplicating or tripling the entire CPU
- Program Analysis approach:
  - Additional memory needed
- Pros of the approach:
  - Easy to realize
  - Less hardware overhead
- Cons of the approach:
  - CPU/firmware based
  - Corrupted memory for checkpoint register
  - Not feasible for AMS circuits





Conclusion and Future Work

- Demanding requirements
- Low power applications
- Production costs
- Approach being considered for industrial use ☺
- Further work for better cost/benefit estimate needed



JOHANNES KEPLER
UNIVERSITY LINZ