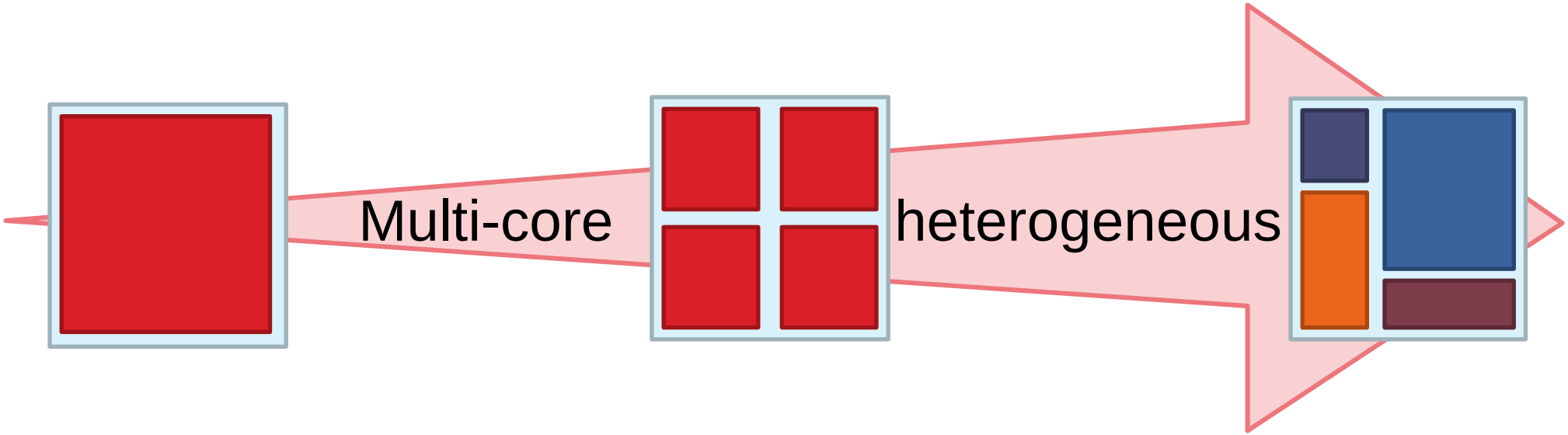
Co-simulation Framework for Streamlining Microprocessor Development on Standard ASIC Design Flow

T. Nakabayashi*, T. Sugiyama*,
T. Sasaki*, E. Rotenberg**,
and T. Kondo*
*Mie University, Japan
**North Carolina State University, USA

Performance gain incurs higher complexity of a processor

A chip design requires a huge effort.



Improving research productivity is essential.

Purpose: boost fabrication of prototype processor

1. Software simulation (e.g., C, C++)

   → Performance estimation using high-level language

2. Register transfer level (RTL)

   → Cycle accurate simulation

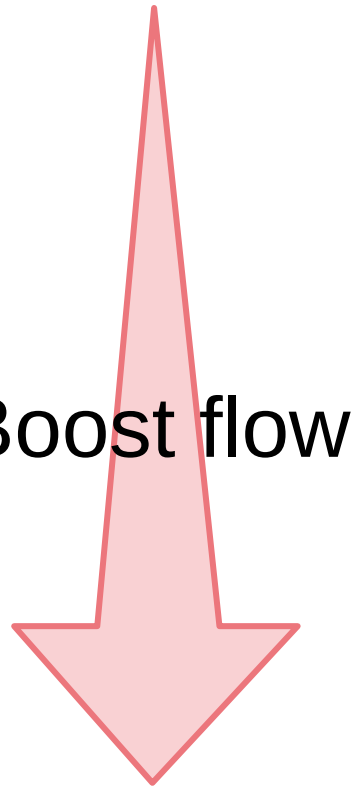
3. Gate level

   → Area, power, delay evaluation

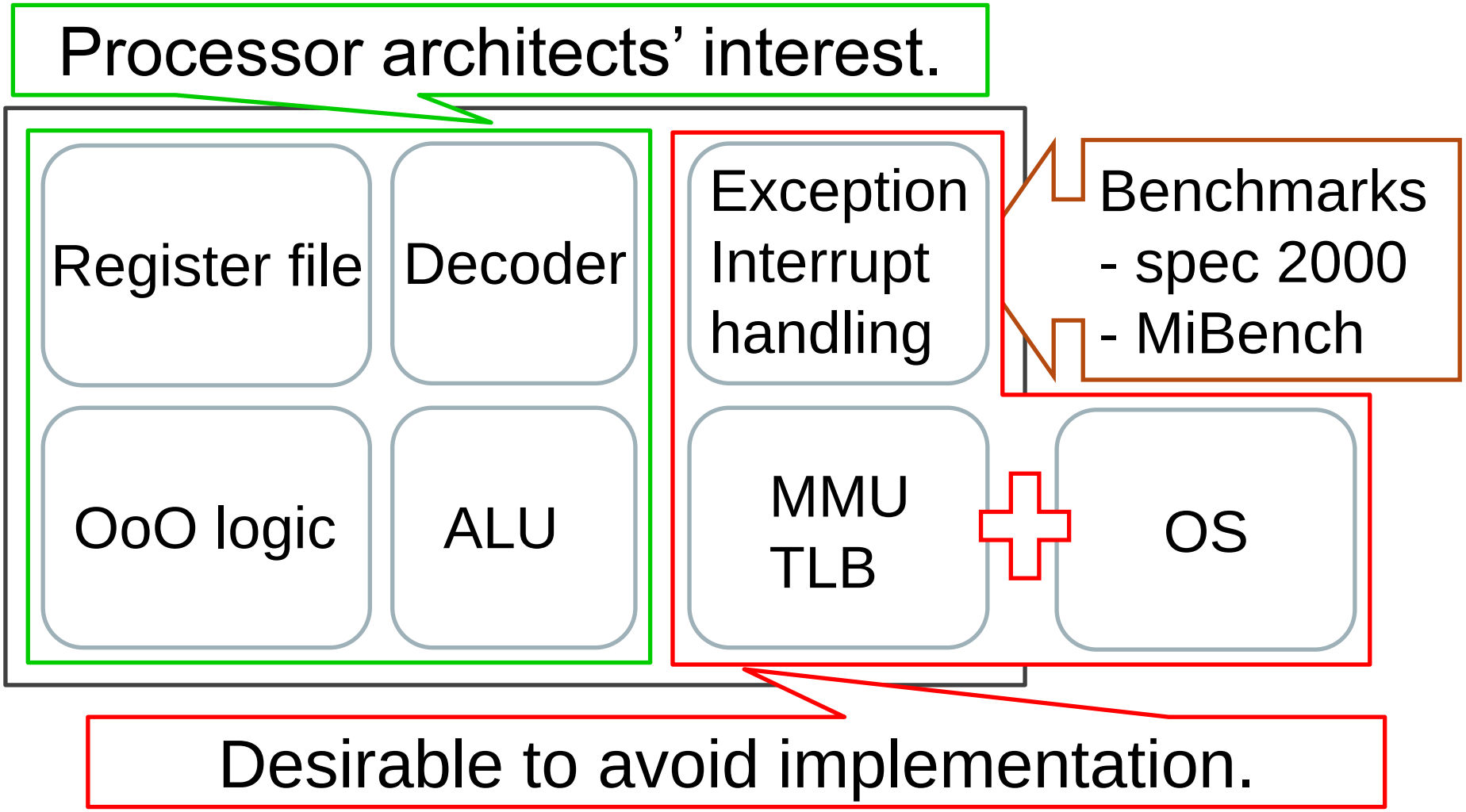
4. Transistor level (post layout)

   → Detailed evaluation

5. Fabrication

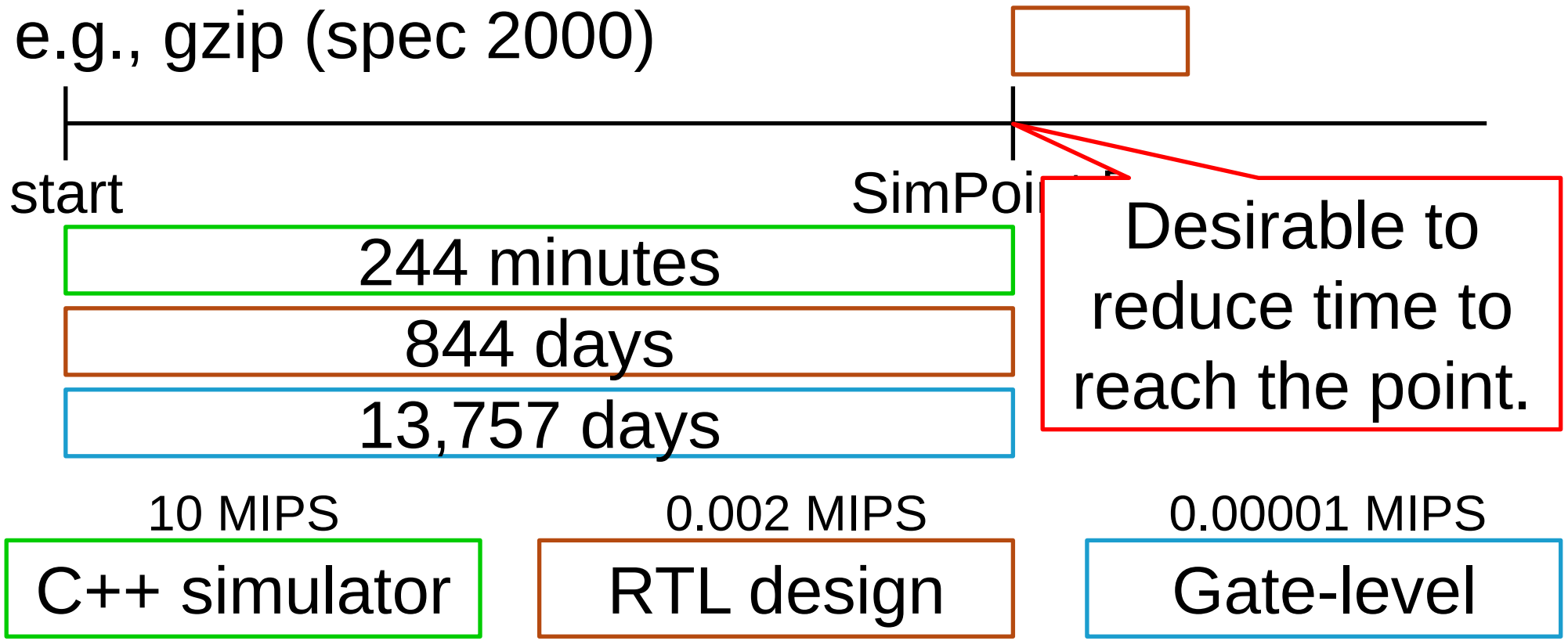


Motivation 1
- Simplify processor prototyping -



Motivation 2
- Reduce simulation time -

Skip initialization or skip to meaningful point.



[1] T. Sherwood, et.al., 10th ASPLOS, 2002

Streamlining mechanism (1/2) - System call emulation -

- Handle standard I/O, file op., network
- System calls are interface with OS kernel

  Privileged mode (MMU, IPR)
  Peripheral circuits (serial port, disk)

in software simulators

system call

Emulate a system call as an one-cycle-instruction

Simplify processor

Streamlining mechanism (2/2)
- Fast skip and state restoration -

For evaluation

we execute a core part of programs due to computational complexity.

Billions of insts. should be forwarded.

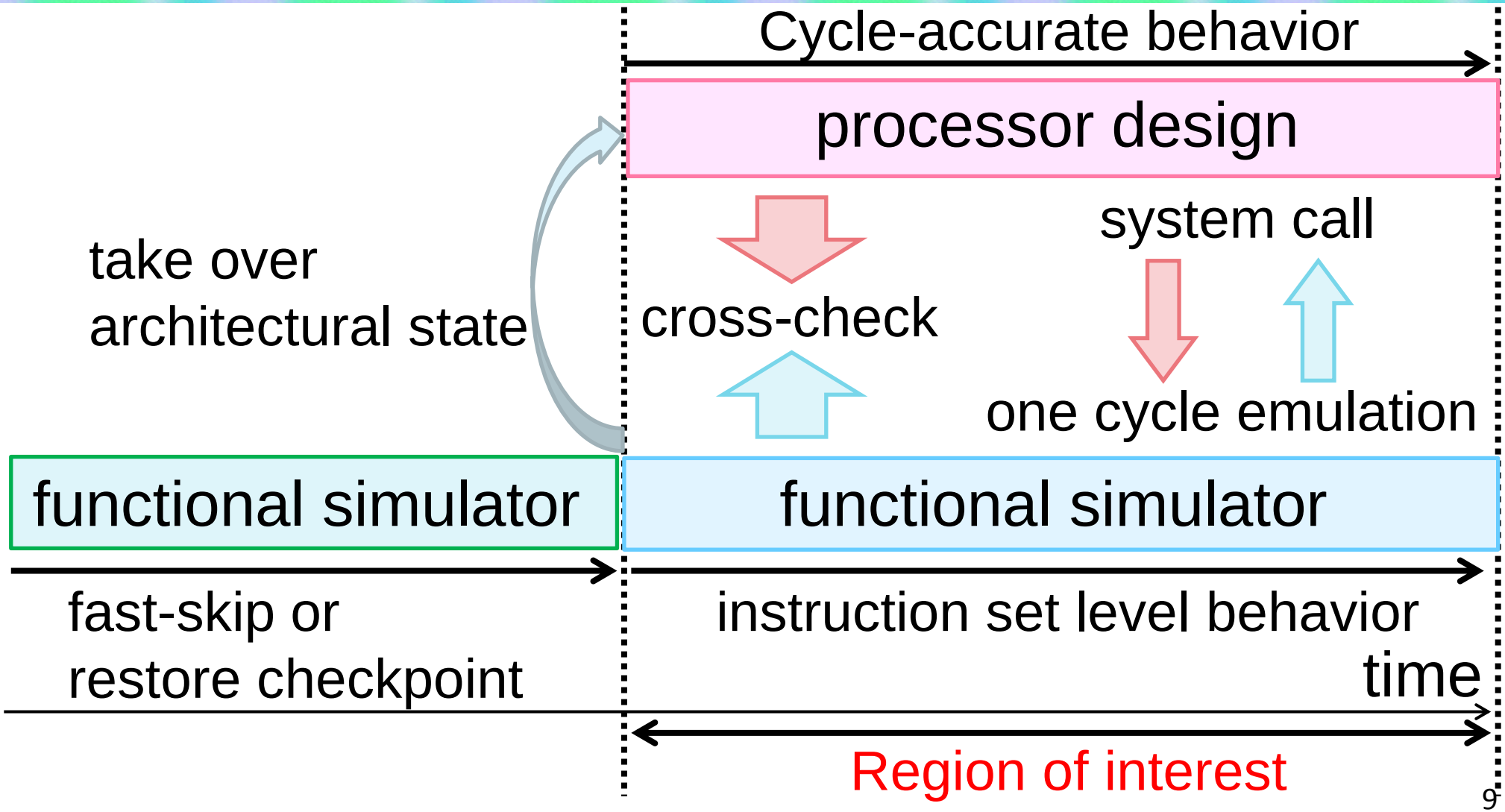
- Software simulators provide
  - fast skip mode and checkpoint mechanism.

Outline

Introduce streamlining mechanisms into standard ASIC design flows.

1. System call emulation
   - introduce an off-chip emulator.
2. Fast skip and state restoration
   - propose a checkpoint mechanism.

Co-simulation framework



Outline

Introduce streamlining mechanisms into standard ASIC design flows.

1. System call emulation

introduce an off-chip emulator.

2. Fast skip and state restoration

propose a checkpoint mechanism.

System call emulation

System call emulation enables

- processor runs programs without OS.
  - Avoid implementation for system kernel and preparation of peripheral devices.

Researchers can focus on microarchitecture.

Challenge on ASIC flows

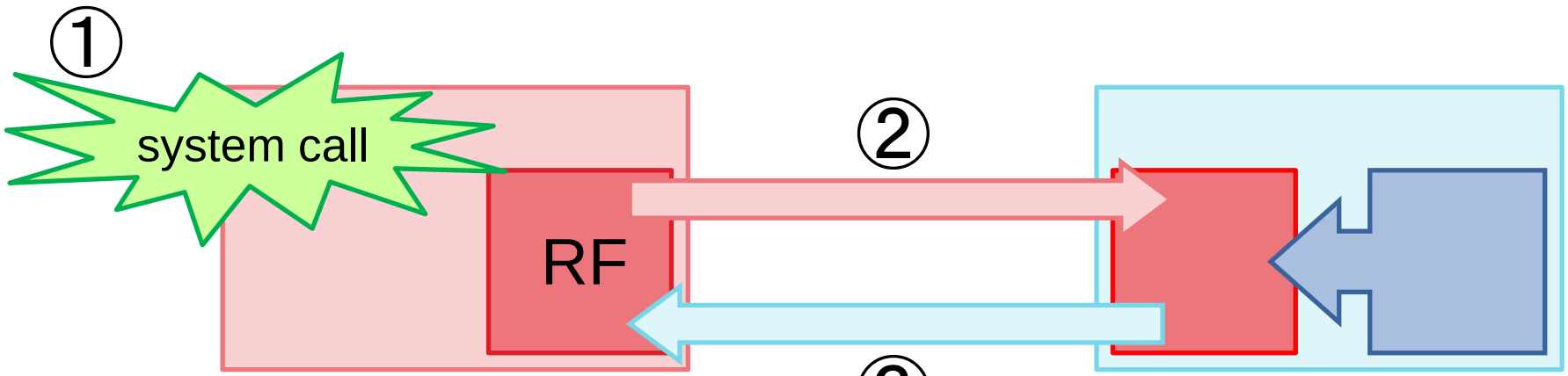
How to interact between processor and emulator.

Sequence of a system call emulation

① emulator must detect the occurrence.

② architectural state (register file) is needed.

③ processor must get feedback of the result.

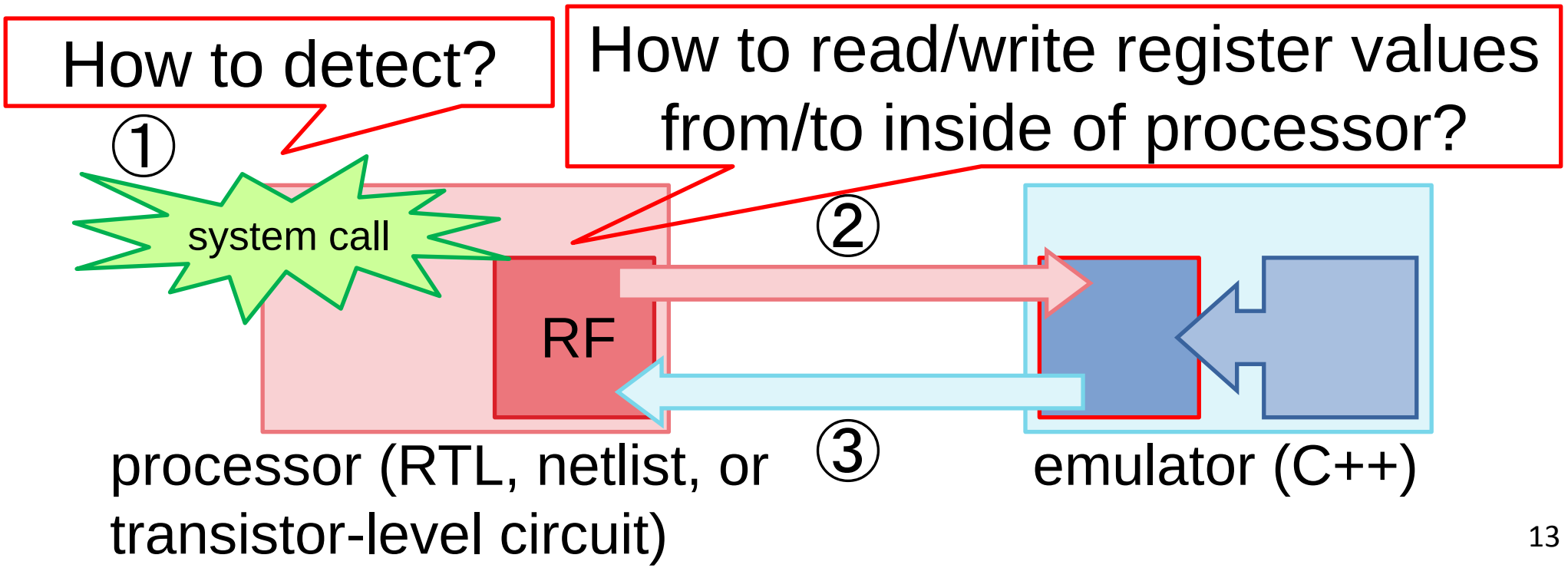


Challenges of a system call emulation

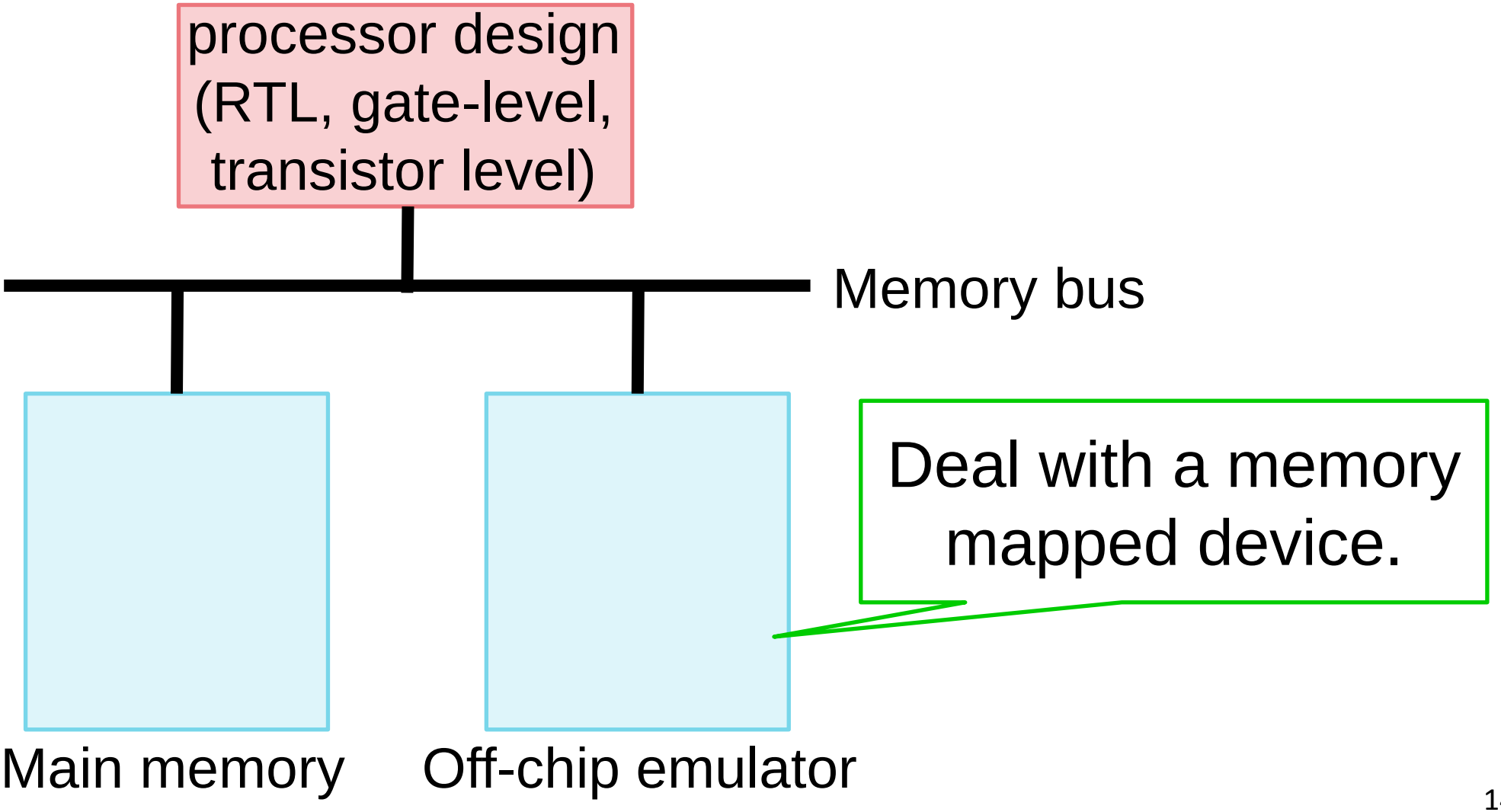
① emulator should know the occurrence.

② architectural state (register file) is needed.

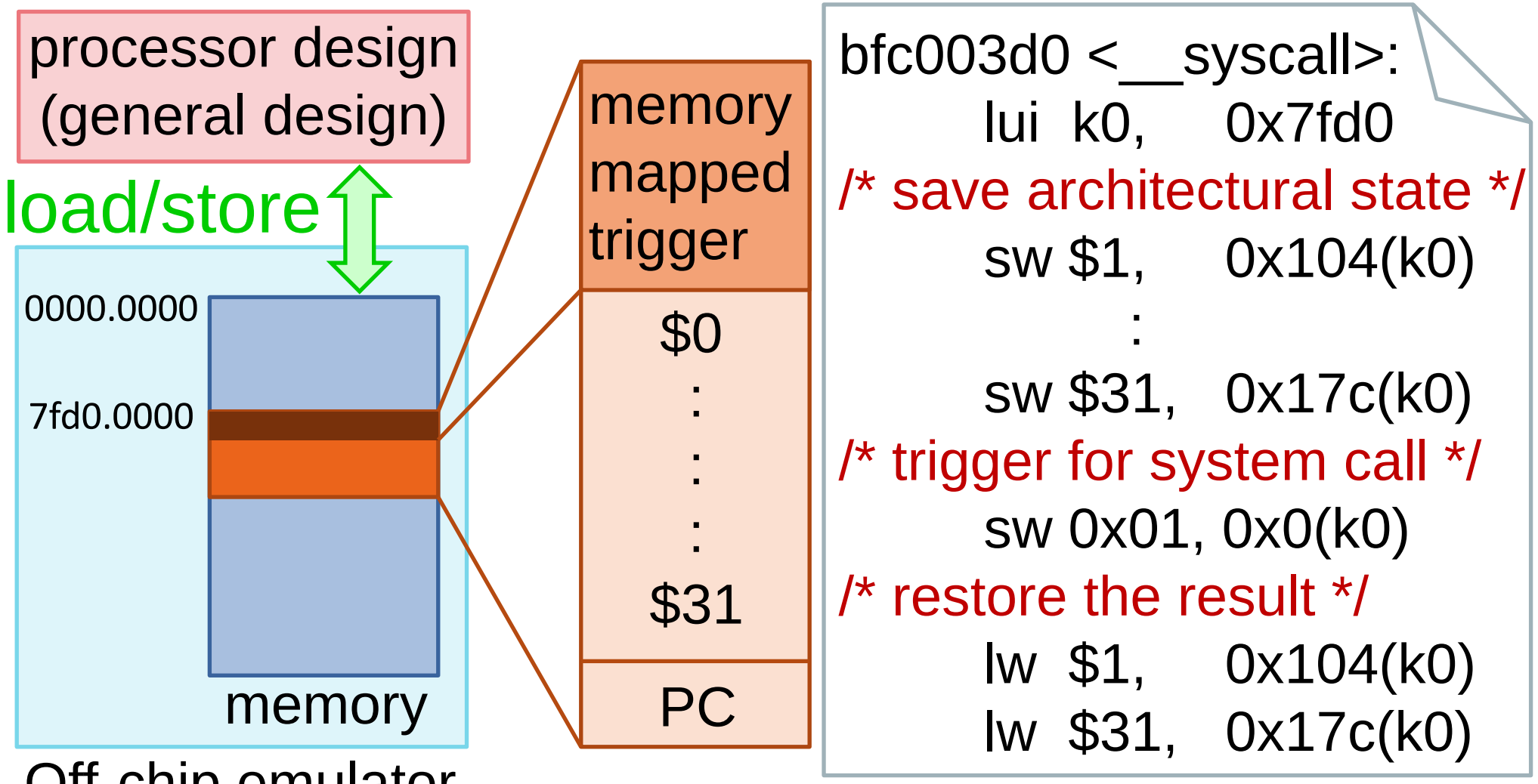
③ processor must get feedback of the result.



Concept of proposed off-chip emulator



Proposed off-chip emulator



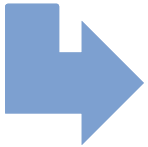
Outline

Introduce streamlining mechanisms into standard ASIC design flows.

1. System call emulation

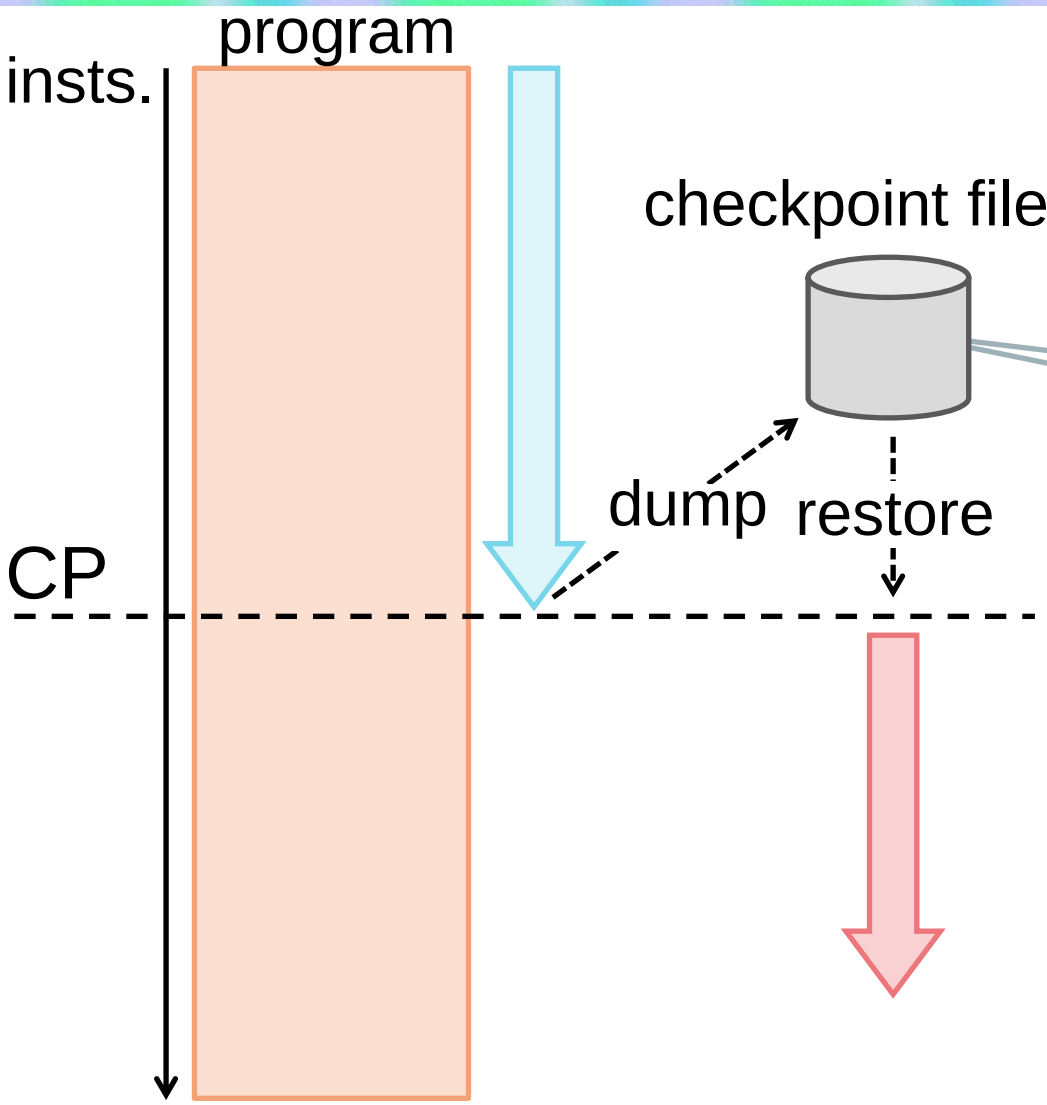
   introduce an off-chip emulator.

2. Fast skip and state restoration

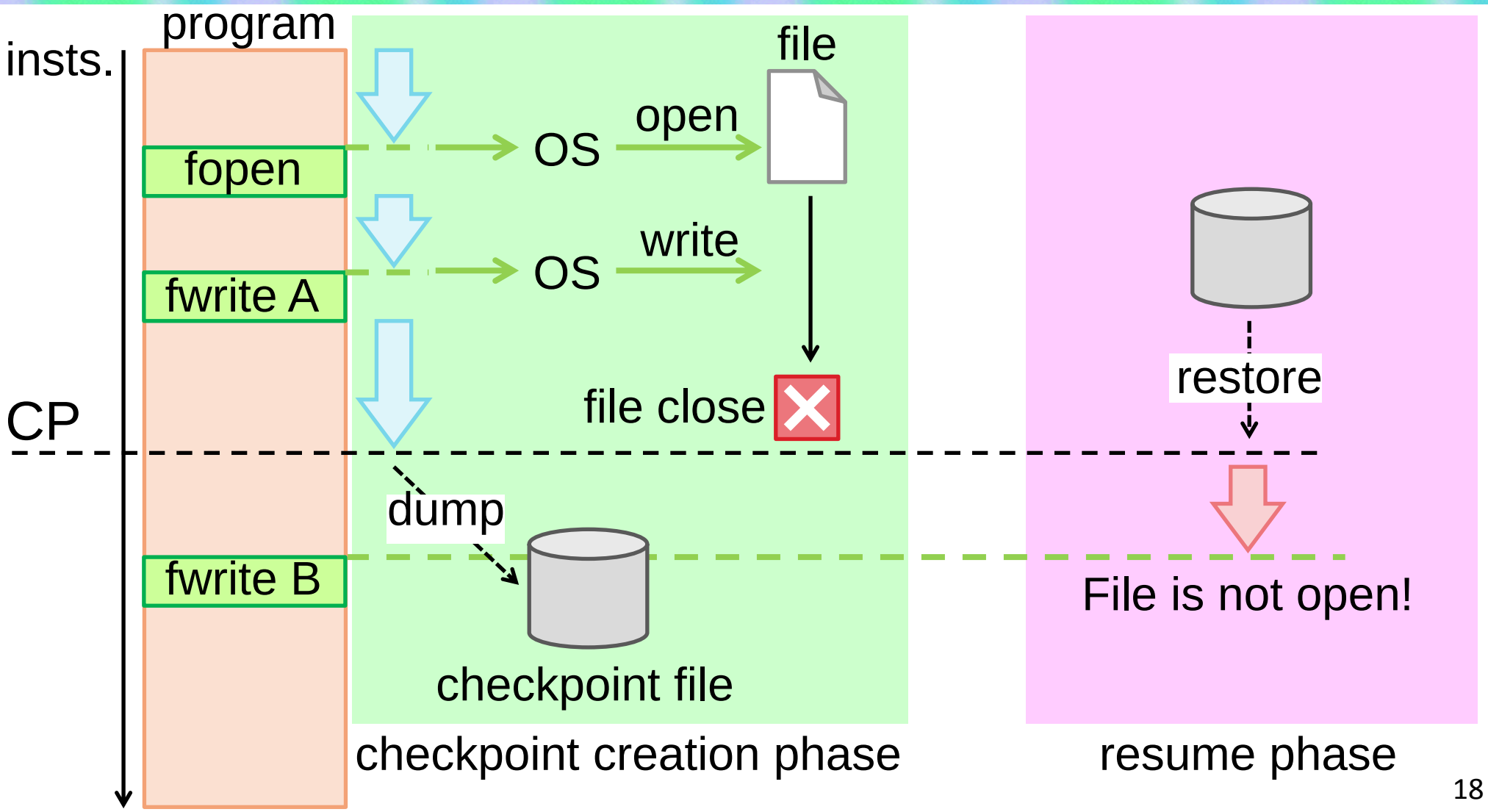
   propose a checkpoint mechanism.

Checkpoint mechanism overview

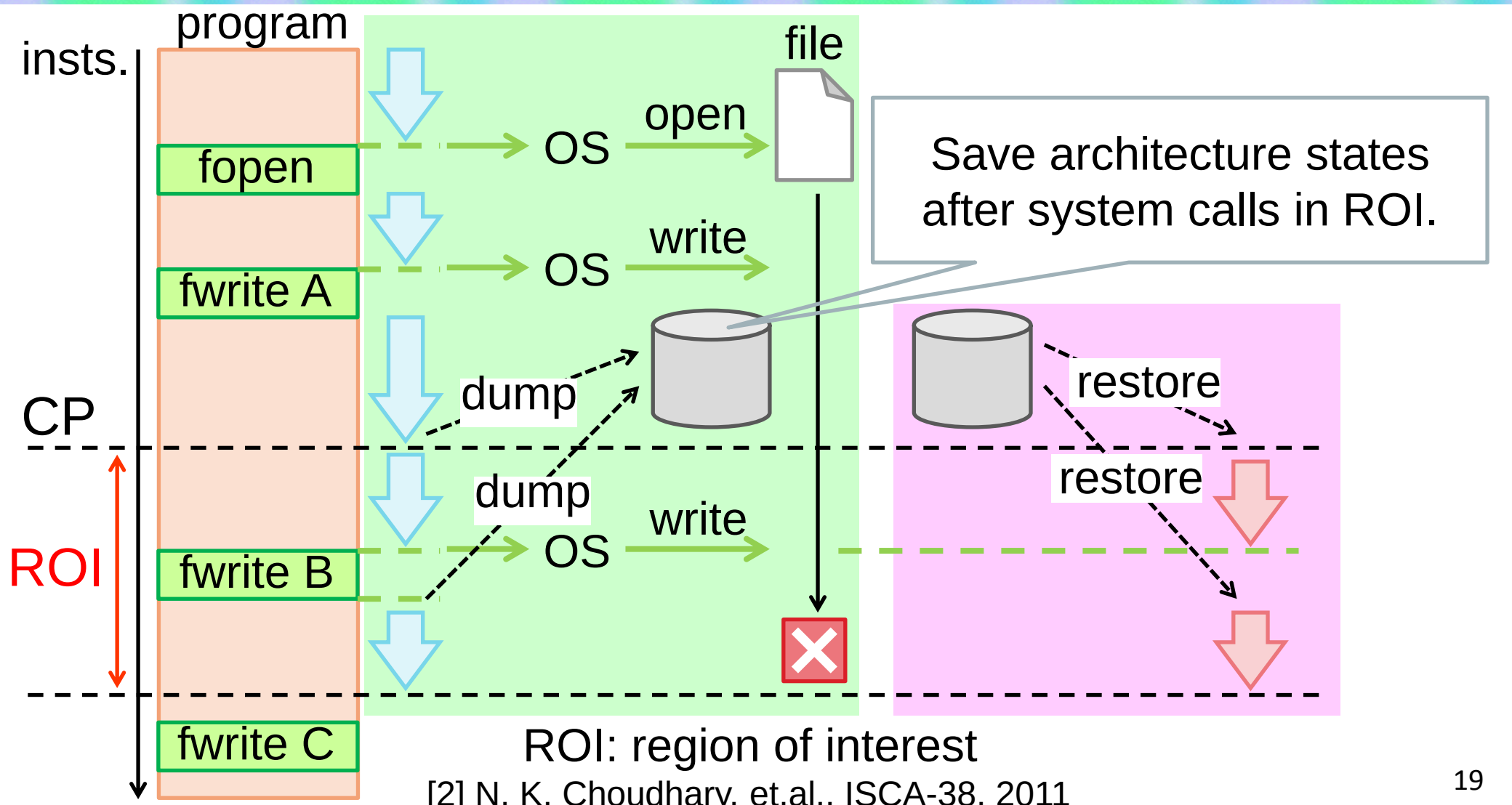



```
bfc00000 <__reset>:
        lui  k0,    0x7fd0
/* restore register file */
        lw  $1,     0x104(k0)
         :   :          :
        lw  $31,    0x17c(k0)
/* restore program counter */
        lw  k1,     0x278(k0)
        jr   k1
```

Challenge of checkpoint



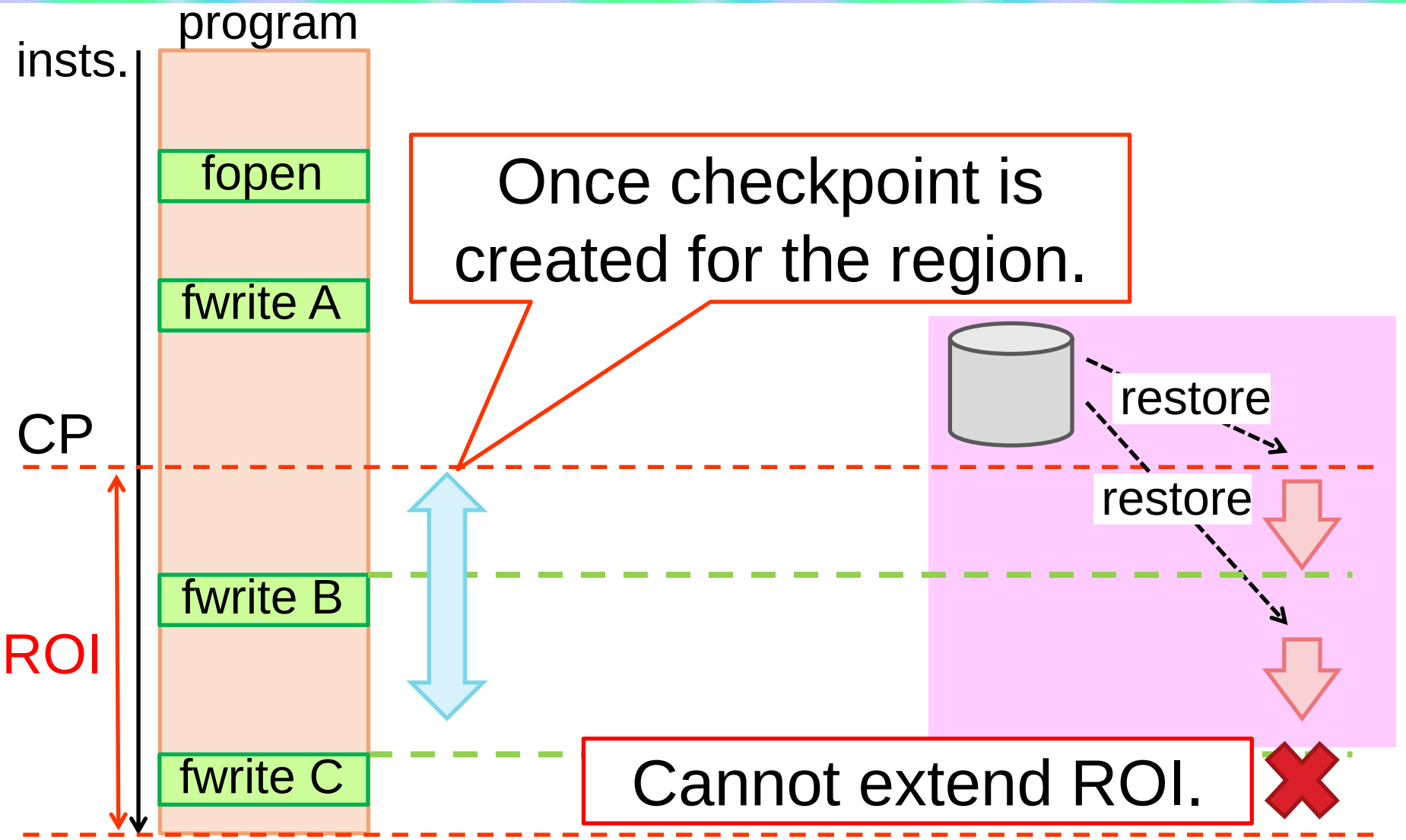
Checkpoint mechanism of FabScalar[2]



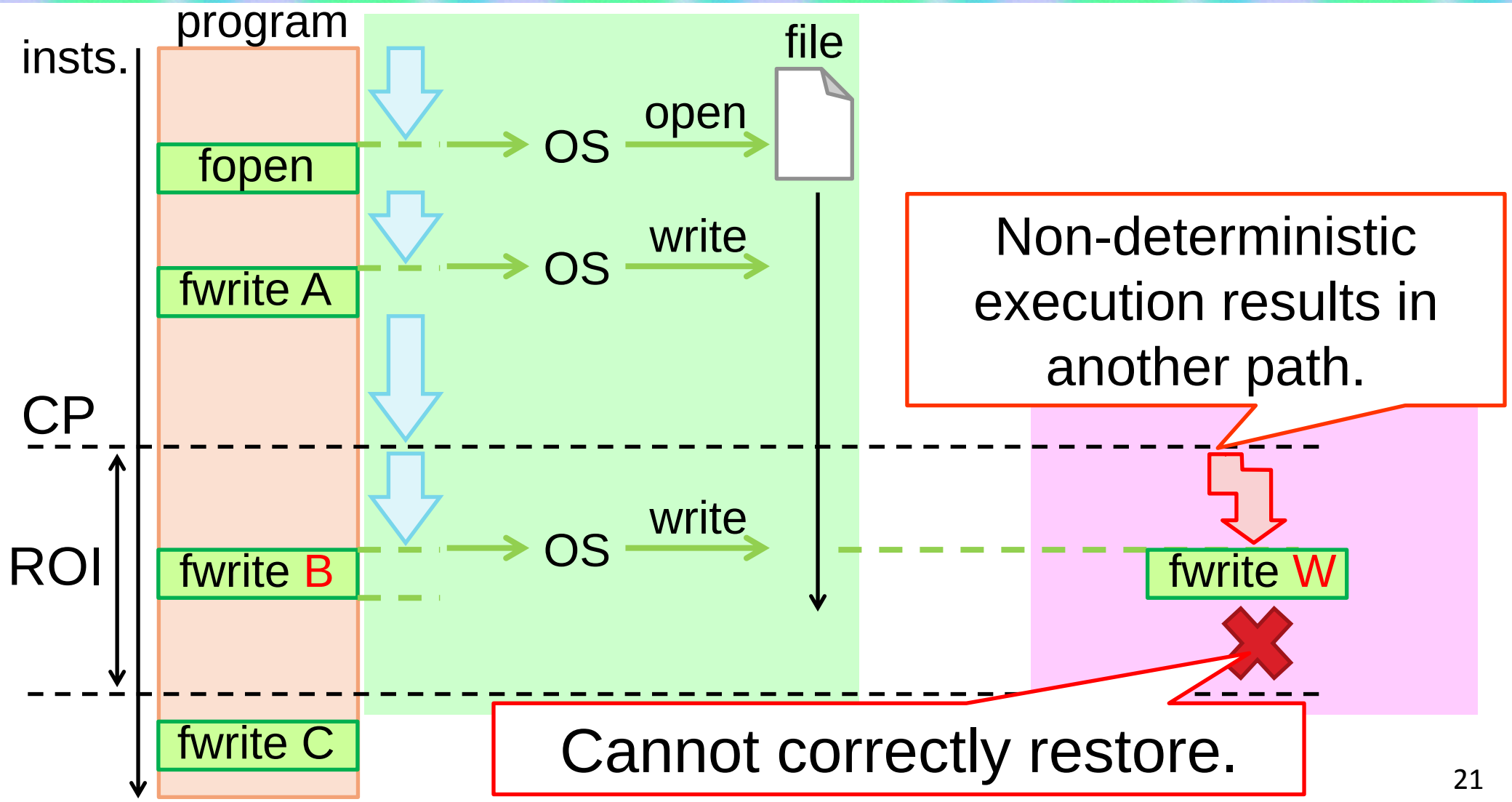
ROI: region of interest

[2] N. K. Choudhary, et.al., ISCA-38, 2011

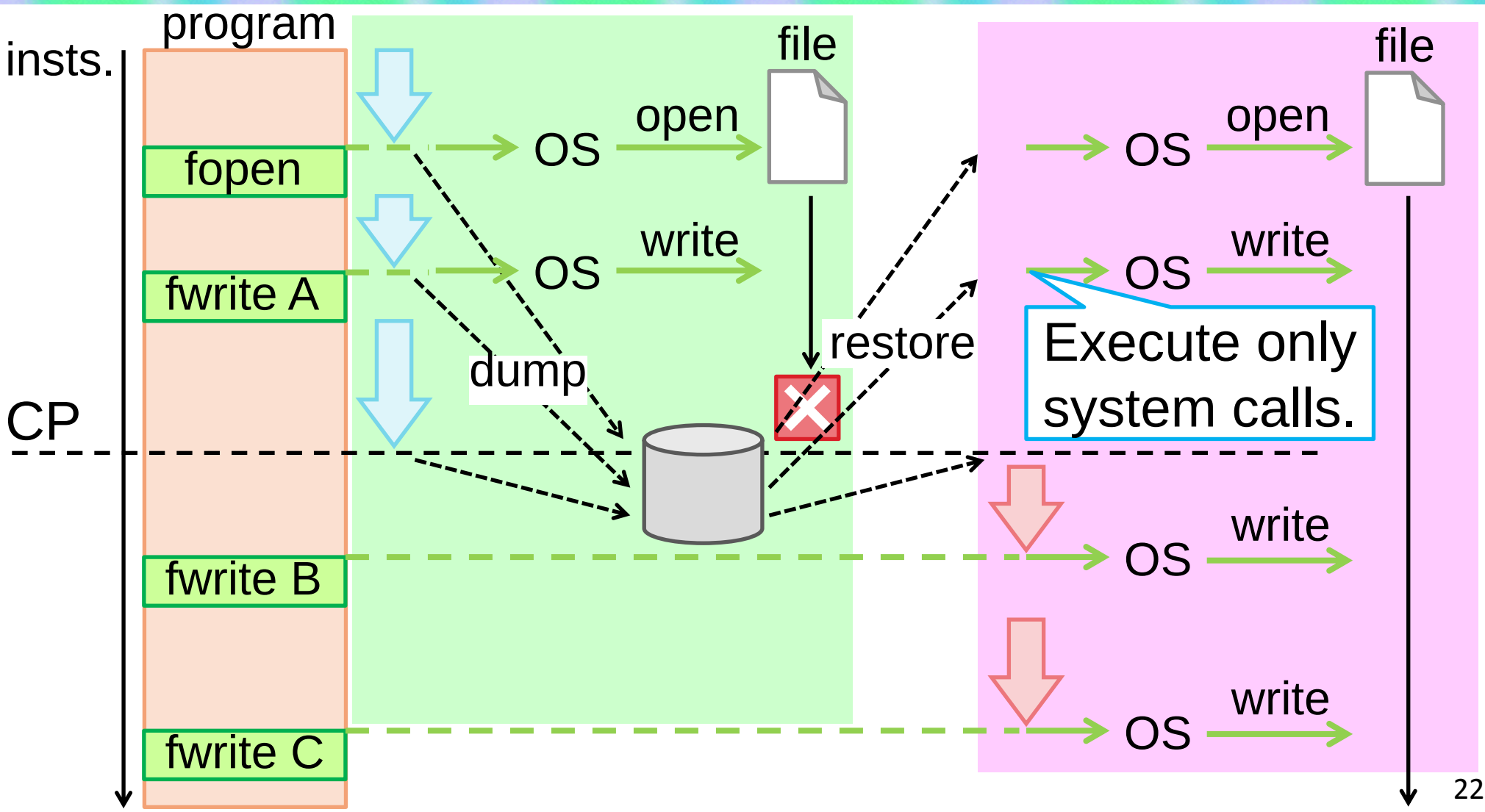
Problems of FabScalar's checkpoint (1/2)



Problems of FabScalar's checkpoint (2/2)



Proposed checkpoint mechanism



Demonstration

Hardware platform: Intel Core i-7-2600@3.40GHz, 4GB memory

	gzip	mcf	bzip	parser	twolf
Gate-level[1] (day)	13,757	6,389	11,297	13,260	12,319
RTL design[1] (day)	844	326	715	680	645
fast-skip (min.)	244	103	206	210	212
checkpoint (sec.)	0.50	0.68	0.77	3.84	0.53
skipped insts. (100 million)	1189	553	977	1146	1066
checkpoint file size (MB)	832	326	384	1780	119
system calls	65	116	101	1027	133

[1]Estimated by million instructions per second (MIPS) value
RTL design is 4-way superscalar generated by FabScalar.

Conclusion

- To boost processor design.
  - Provide off-chip system call emulator.
    - Execute programs without booting OS.
  - Introduce checkpoint into ASIC flows.
    - Shorten turnaround time.
- Future work
  - Demonstration using fabricated chip.
  - The entire framework will be open.