DATE 17
DESIGN, AUTOMATION & TEST IN EUROPE
27 – 31 March, 2017 · STCC · Lausanne · Switzerland
The European Event for Electronic
System Design & Test

Hardware-Accelerated Dynamic Binary Translation

Rokicki Simon - Irisa / Université de Rennes 1

Steven Derrien - Irisa / Université de Rennes 1

Erven Rohou - Inria

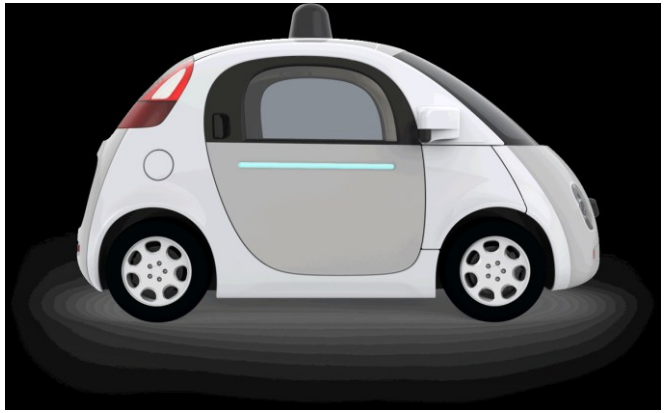
UMR IRISA

Inria
INVENTORS FOR THE DIGITAL WORLD

Embedded Systems

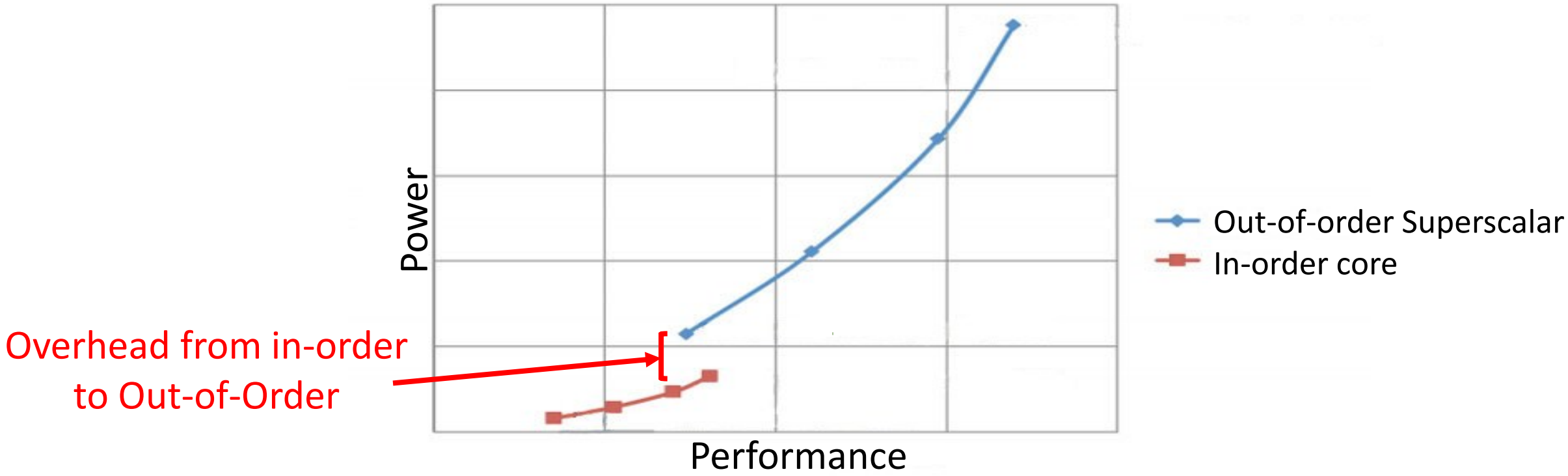
Tight constraints in

- Power consumption
- Production cost
- Performance



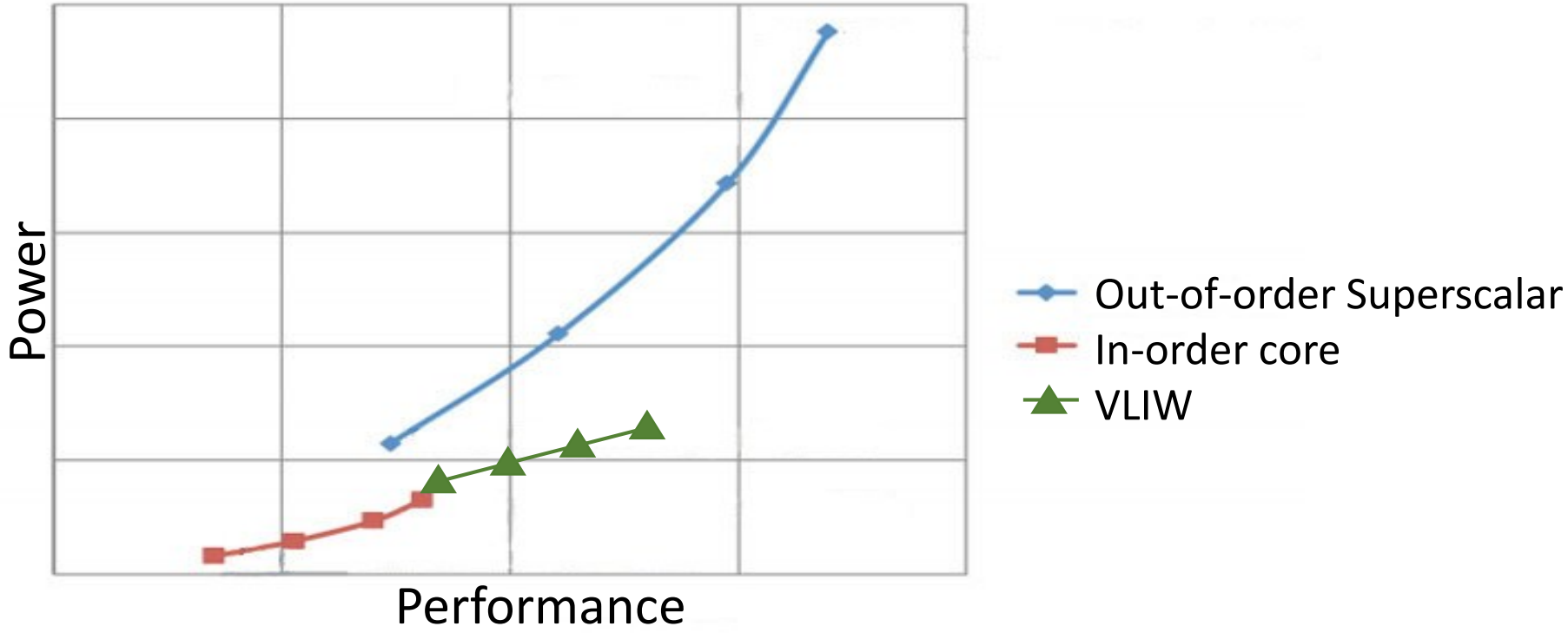
Systems on a Chip

- Complex heterogeneous designs
- Heterogeneity brings new power/performance trade-off

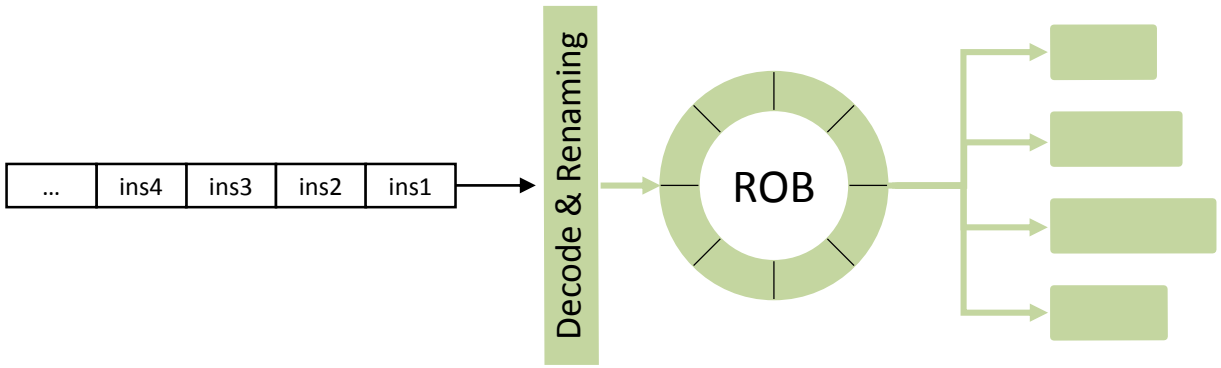


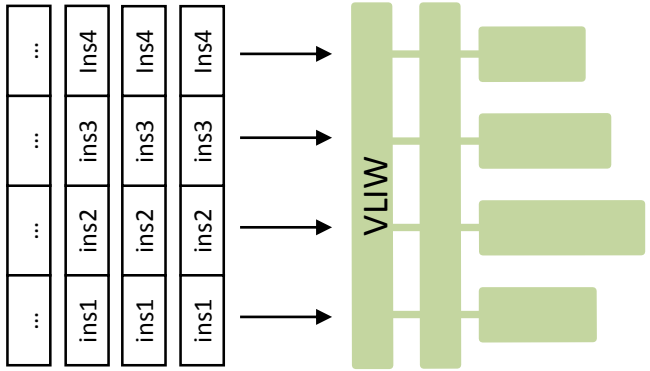
Systems on a Chip

- Complex heterogeneous designs
- Heterogeneity brings new power/performance trade-off
- **Are there better trade-off?**



Architectural choice





Out-of-Order processor

- Dynamic Scheduling
- **Performance portability**
- Poor energy efficiency

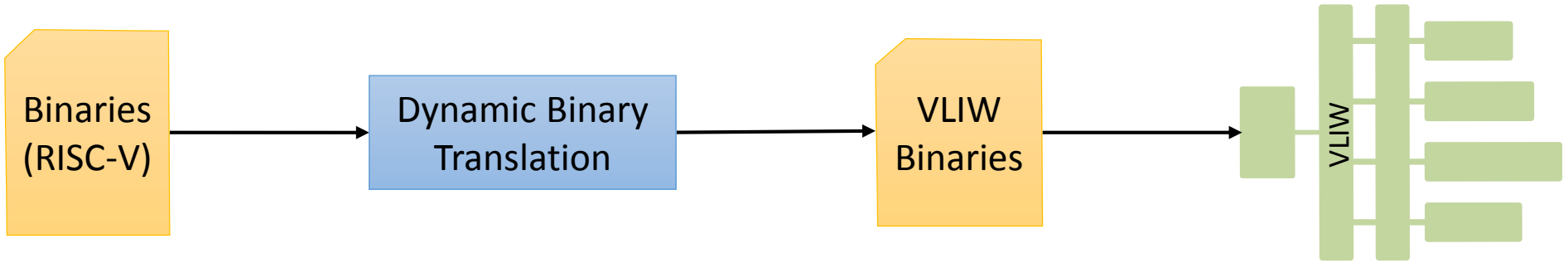
VLIW processor

- Static scheduling
- No portability
- **High energy efficiency**

The best of both world ?

Dynamically translate native binaries into VLIW binaries:

- Performance close to Out-of-Order processor
- Energy consumption close to VLIW processor



Existing approaches

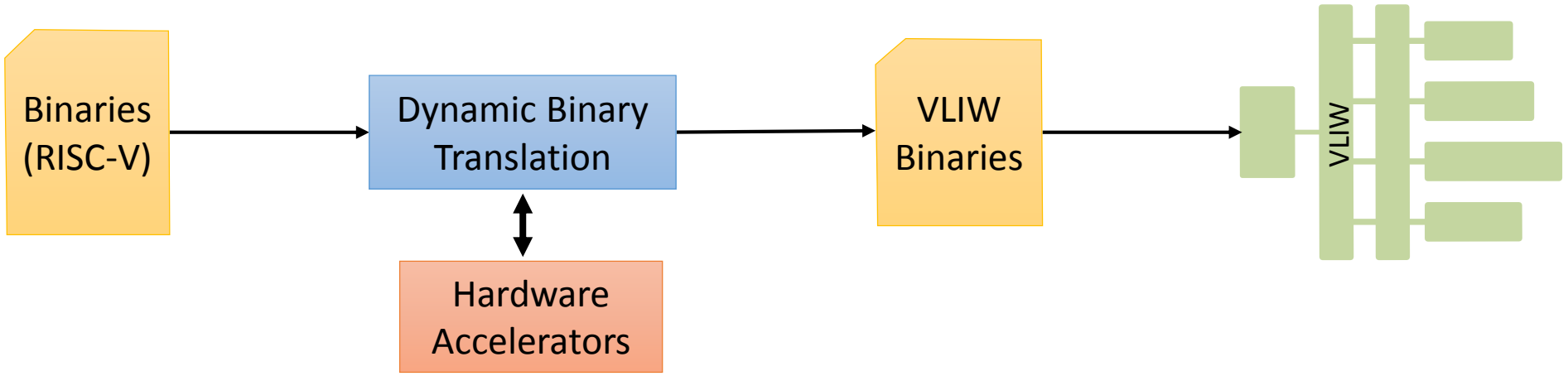
- Transmeta Code Morphing Software & Crusoe architectures
  - x86 on VLIW architecture
  - User experience polluted by cold-code execution penalty

- Nvidia Denver architecture
  - ARM on VLIW architecture

- **Translation overhead is critical**
- **Too few information on closed platforms**

Our contribution

- Hardware accelerated DBT framework
  - **Make the DBT cheaper (time & energy)**
  - **First approach that try to accelerate binary translation**
- Open source framework
  - **Allows research**



Outline

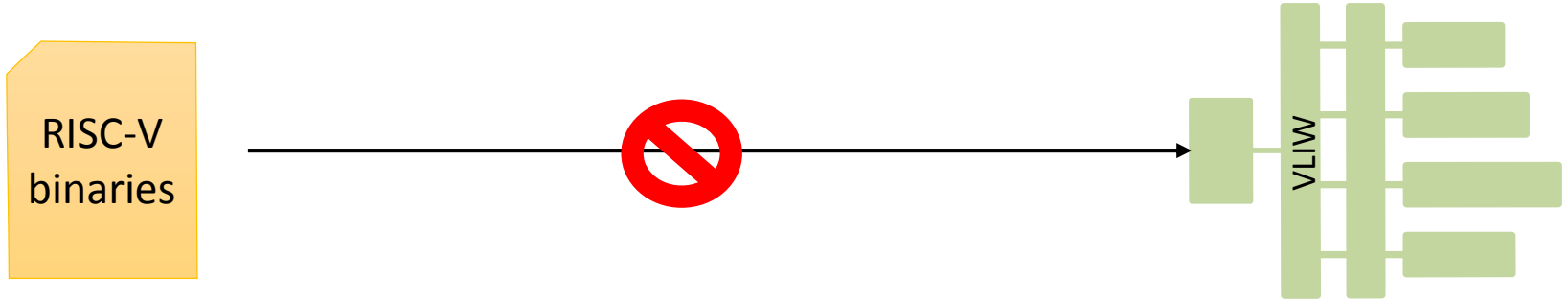
- Hybrid-DBT Platform
  - How does it work?
  - What does it cost?
  - Focus on optimization levels
- Experimental Study
  - Impact on translation overhead
  - Impact on translation energy overhead
  - Impact on area utilization
- Conclusion & Future work

Outline

- **Hybrid-DBT Platform**
  - **How does it work?**
  - **What does it cost?**
  - **Focus on optimization levels**
- Experimental Study
  - Impact on translation overhead
  - Impact on translation energy overhead
  - Impact on area utilization
- Conclusion & Future work

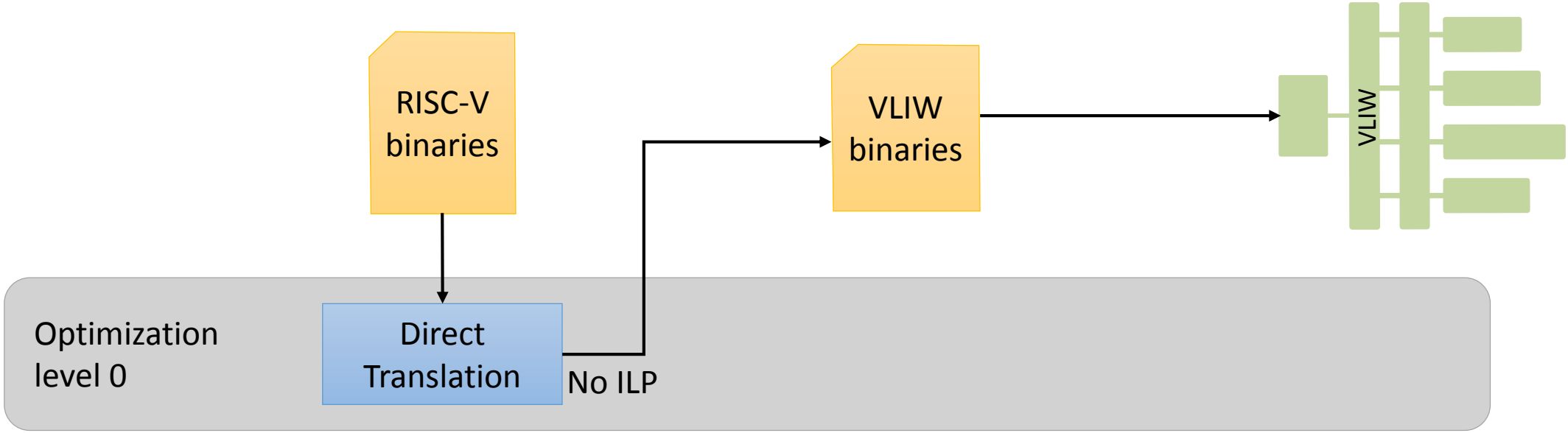
How does it work?

- RISC-V binaries cannot be executed on VLIW



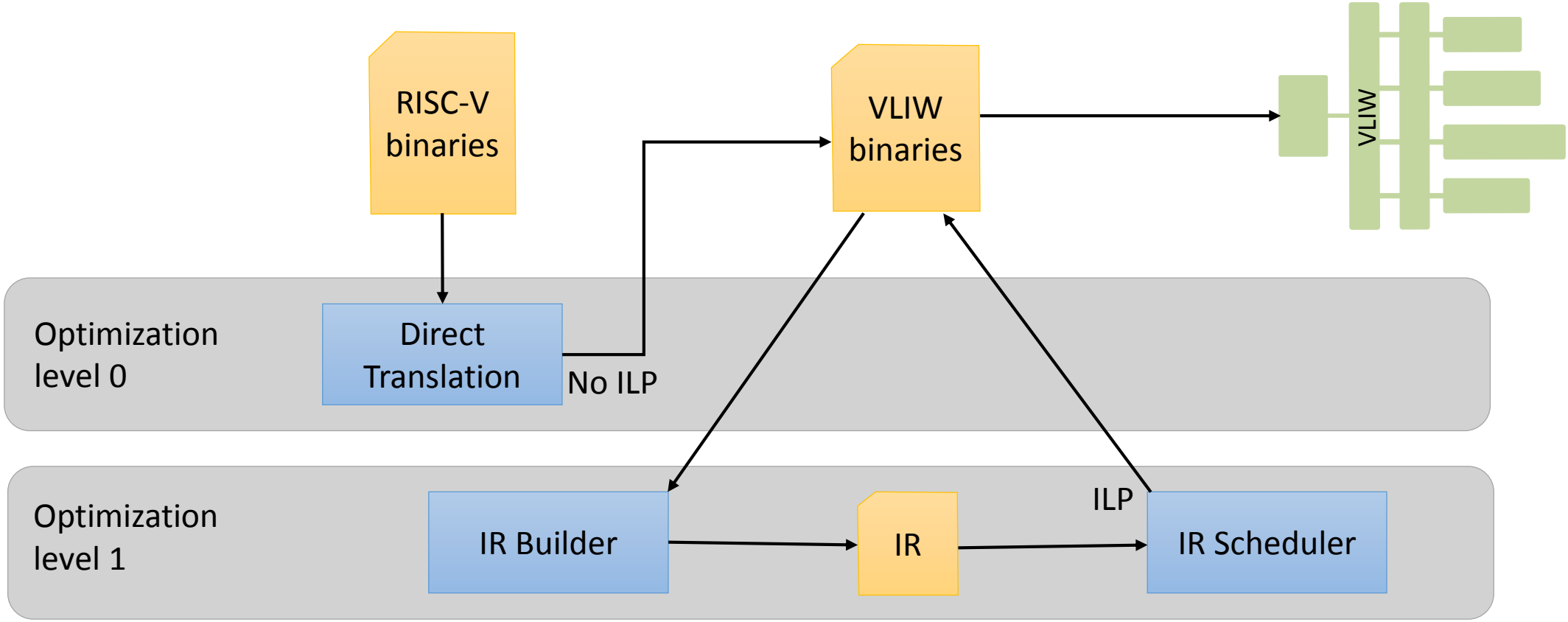
How does it work?

- Direct, naive translation from native to VLIW binaries
- Does not take advantage of Instruction Level Parallelism



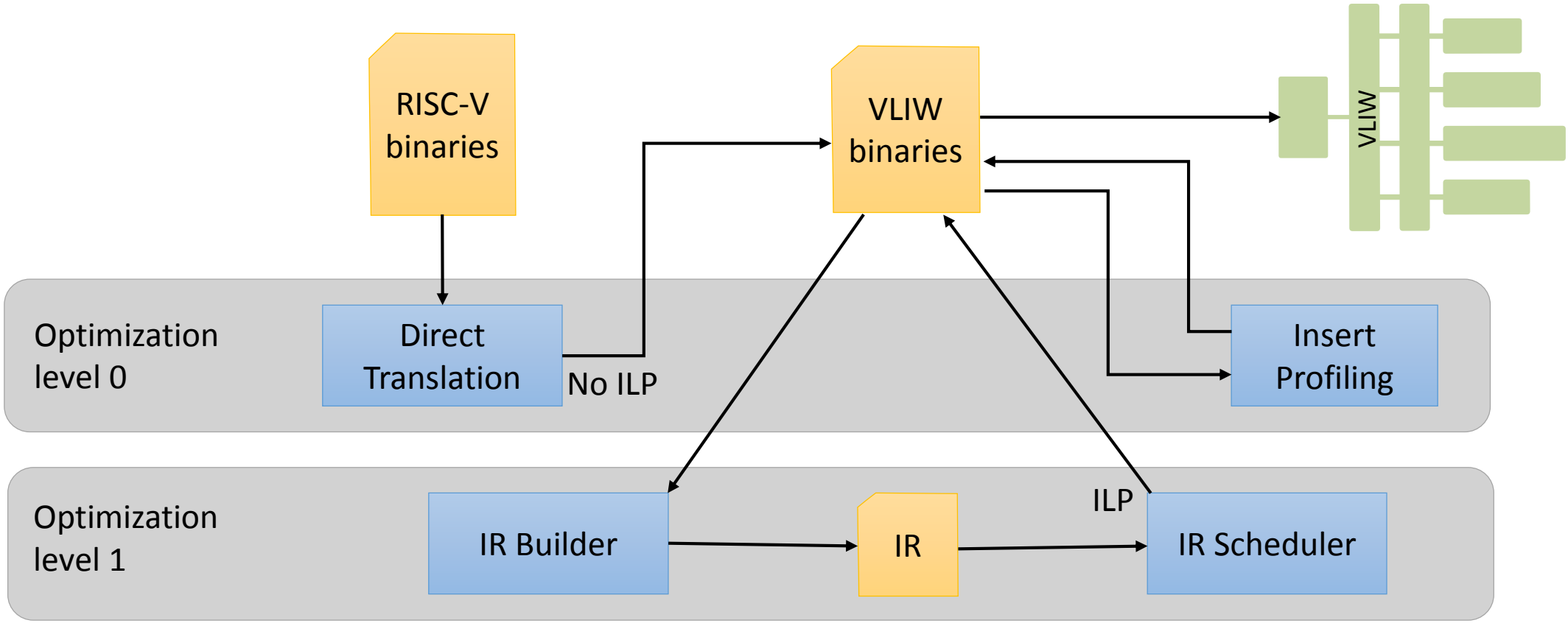
How does it work?

- Build an Intermediate Representation (CFG + dependencies)
- Reschedule Instructions on VLIW execution units



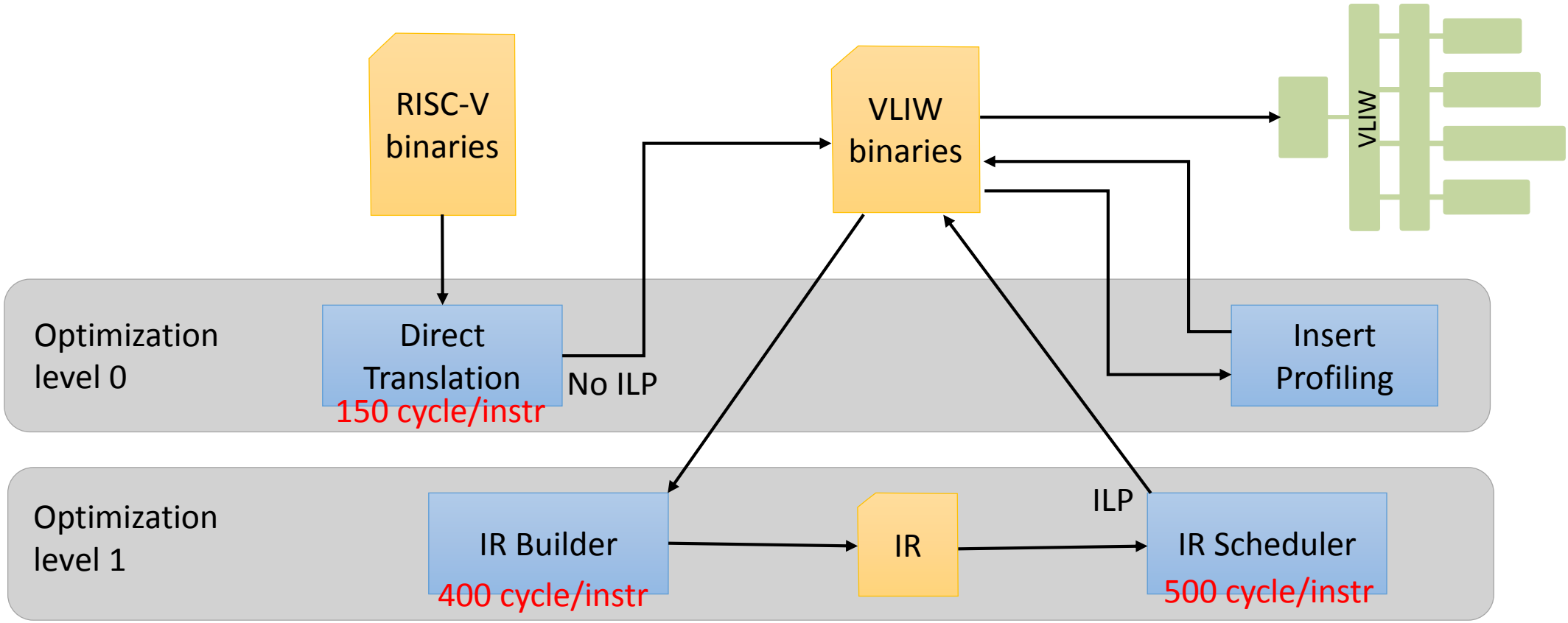
How does it work?

- Code profiling to detect hotspot
- Optimization level 1 only on hotspots



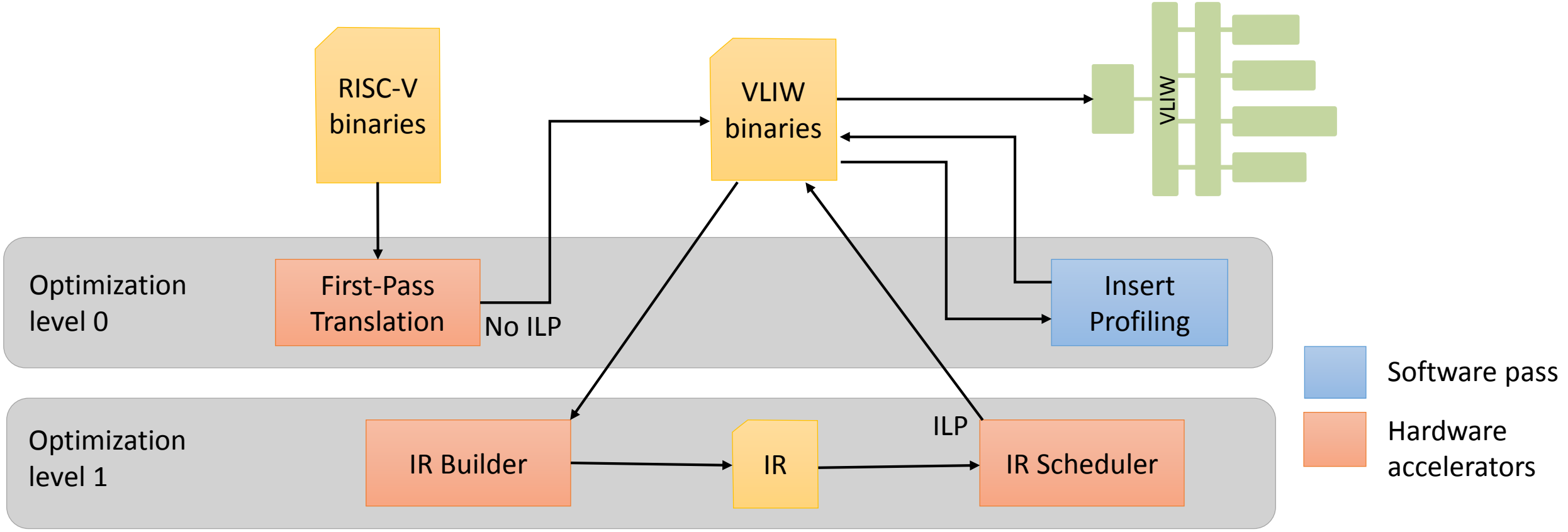
What does it cost?

- Cycle/instr : number of cycles to translate one RISC-V instruction
- **Need to accelerate time consuming parts of the translation**

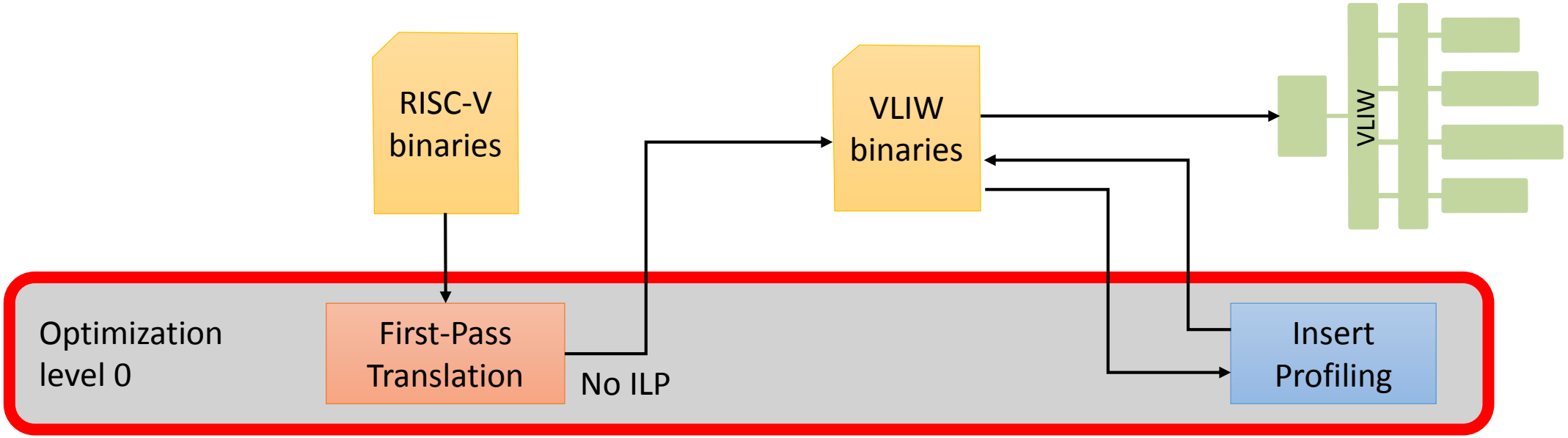


Hybrid-DBT framework

- **Hardware acceleration of critical steps of DBT**
  - **Can be seen as a hardware accelerated compiler back-end**



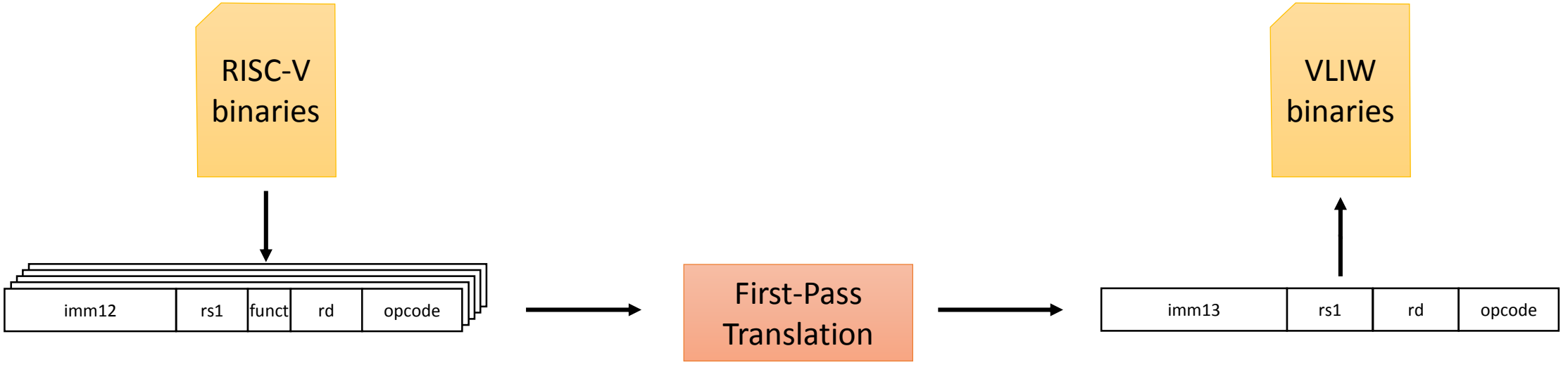
Focus on optimization level 0



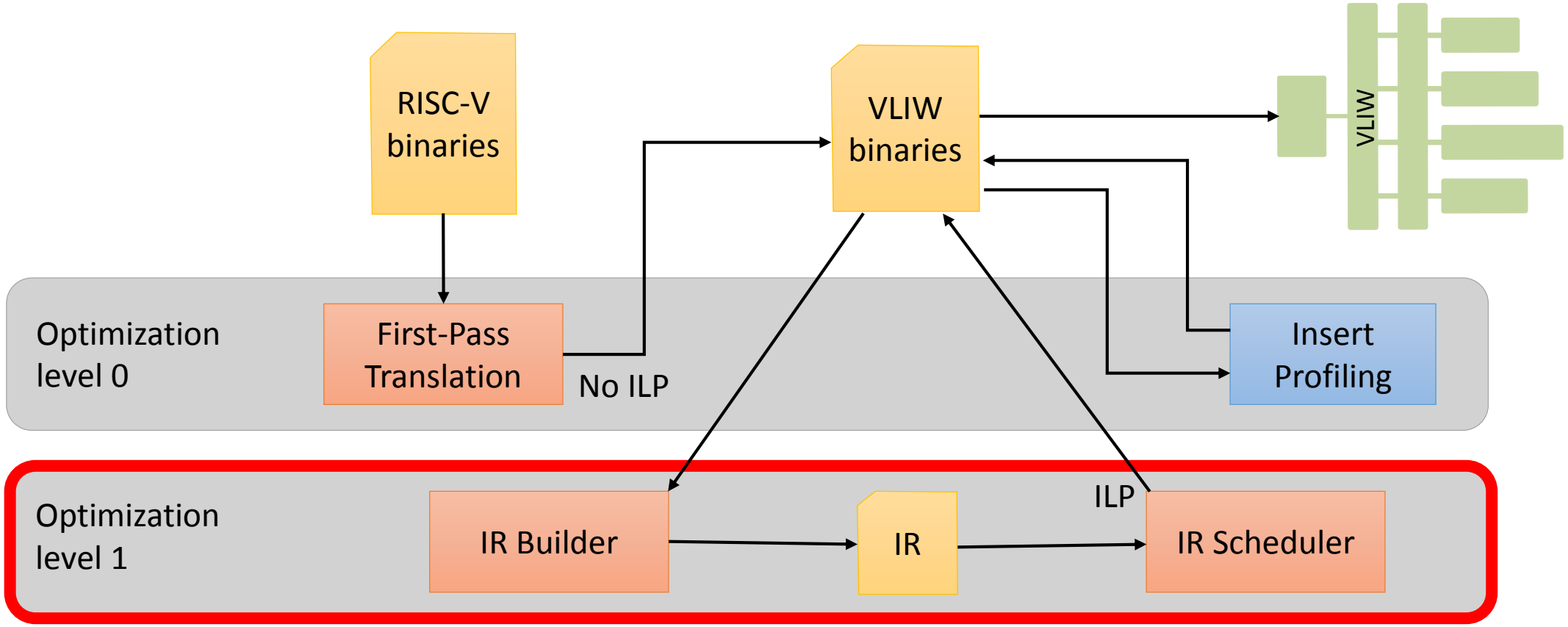
- Critical for system reactivity

First-Pass Translation

- Implemented as a Finite State Machine
  - Translate each native instruction separately
  - Produces 1 VLIW instruction per cycle
  - 1 RISC-V instruction => up to 2 VLIW instructions
- Simple because ISA are similar



Focus on optimization level 1



- Critical to start exploiting VLIW capabilities

Goal of optimization level 1

Native Binaries

```
stw  r5,0(r3)
ldw  r3,0(r2)
addi r4,r1,1
sub  r4,r4,r3
stw  r4,0(r3)
movi r3,0
```


Exploit available ILP

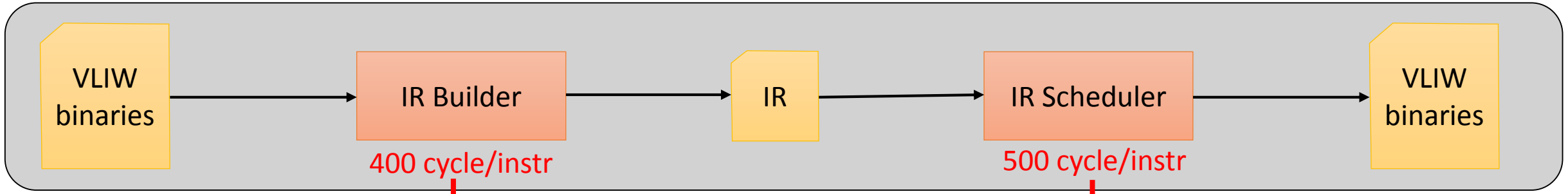
- Compute dependencies
- Perform Instruction Scheduling

VLIW Binaries

```
nop              stw r5,0(r3)
addi r4,r1,1     ldw r3,0(r2)
sub r4,r4,r3     nop
movi r3,0        stw r4,0(r3)
```

Cost of optimization level 1

- Generate high-level IR
- Instruction scheduling on the IR



acceleration is simple

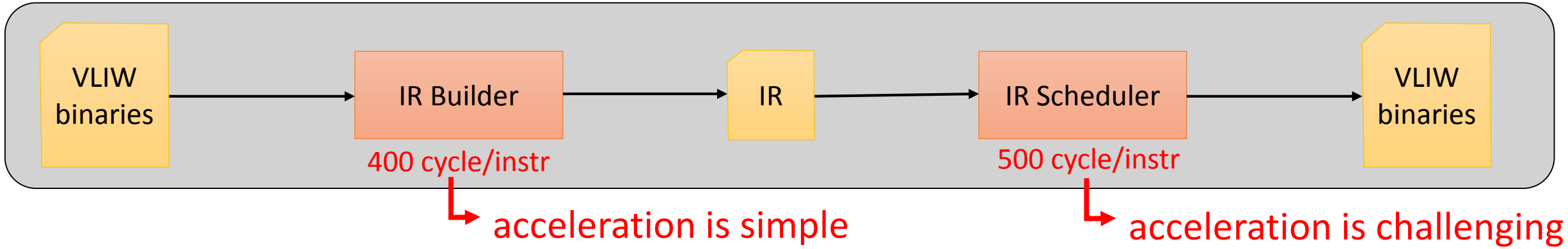
- Instruction decoding/encoding
- Single FOR loop
- Regular computations

acceleration is challenging

- Difficult to parallelize
- Complex control flow structure

Cost of optimization level 1

- Generate high-level IR
- Instruction scheduling on the IR

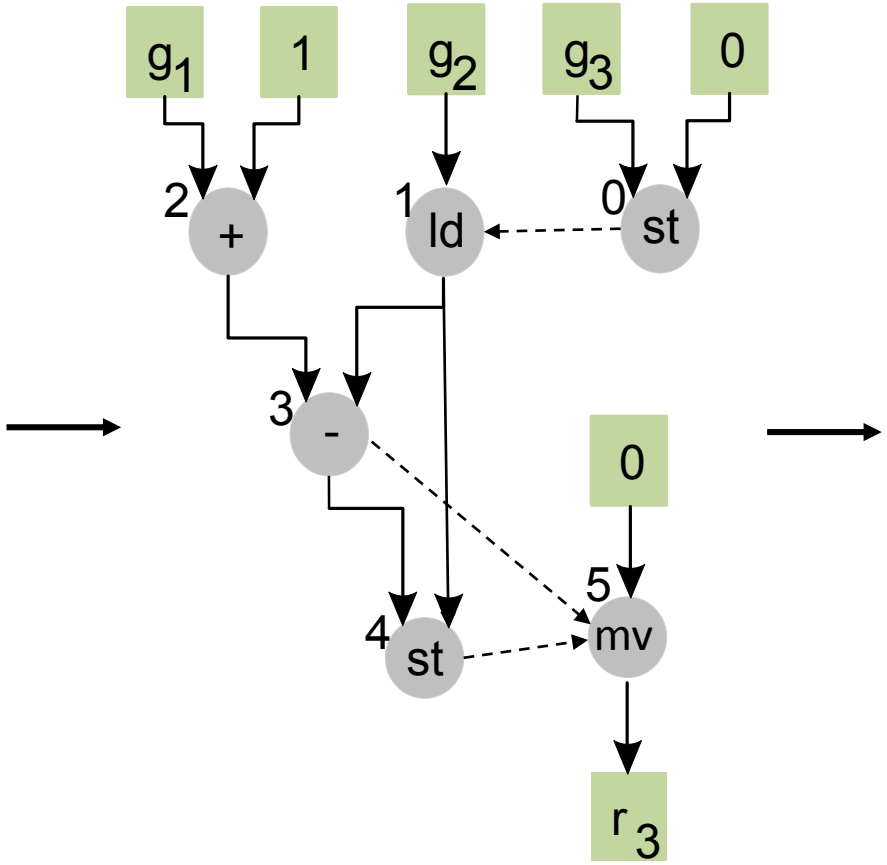


- Instruction Scheduling is the bottleneck
  - **IR is designed to speed-up scheduling**

Choosing an Intermediate Representation

Native Binaries

```
stw   r5,0(r3)
ldw   r3,0(r2)
addi  r4,r1,1
sub   r4,r4,r3
stw   r4,0(r3)
movi  r3,0
```



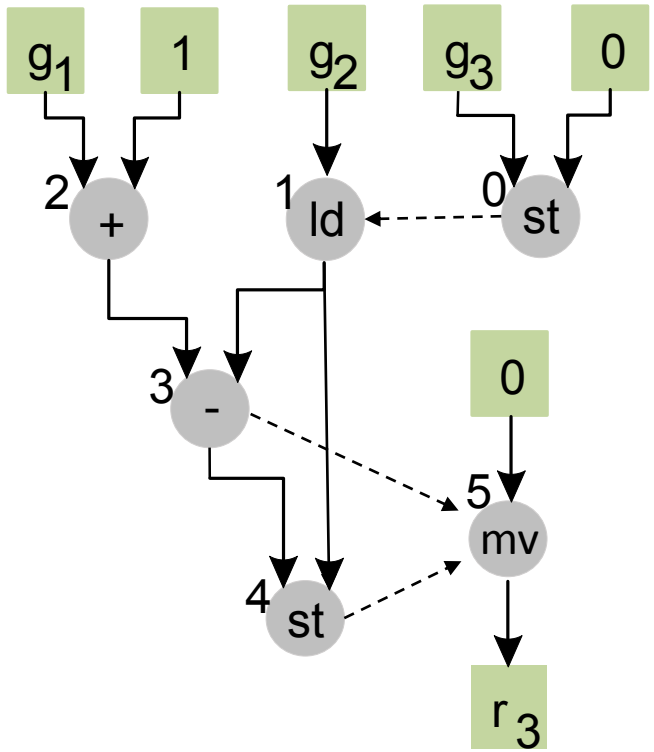
VLIW Binaries

```
nop                stw r5,0(r3)
addi r4,r1,1       ldw r3,0(r2)
sub r4,r4,r3       nop
movi r3,0          stw r4,0(r3)
```

Choosing an Intermediate Representation

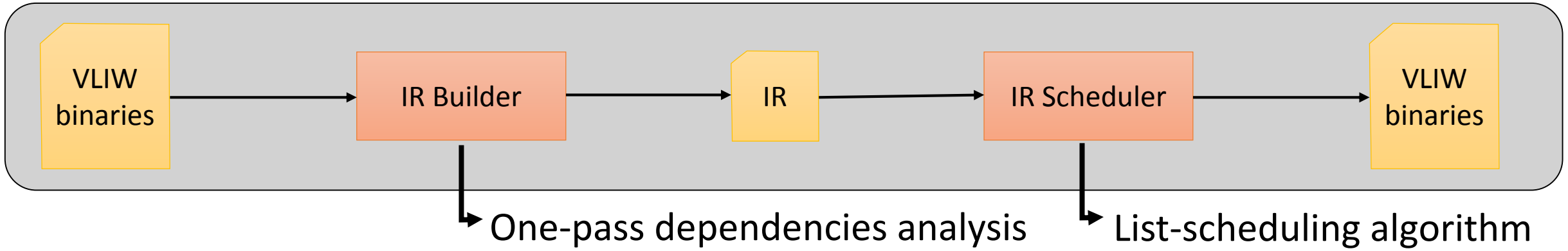
IR advantages:

- Direct access to dependencies and successors
- Regular structure (no pointers / variable size)



op	registers[4]	nbDep	nbDSucc	nbSucc	succNames[8]
0-st	@g3 = 0	0	0	1	1 - - - - - - -
1-ld	r1 = @g2	1	2	2	3 4 - - - - - -
2-addi	g1 = g1 1	0	1	1	3 - - - - - - -
3-sub	r3 = r1 g1	2	1	2	4 5 - - - - - -
4-st	@g2 = r3	2	0	1	5 - - - - - - -
5-mov	r3 = 0	2	0	0	- - - - - - - -

Details on hardware accelerators



- **Developing such accelerators using VHDL is out of reach**
- Accelerators are developed using High-Level Synthesis
  - Loops unrolling/pipelining
  - Memory partitioning
  - Memory accesses factorization
  - Explicit forwarding

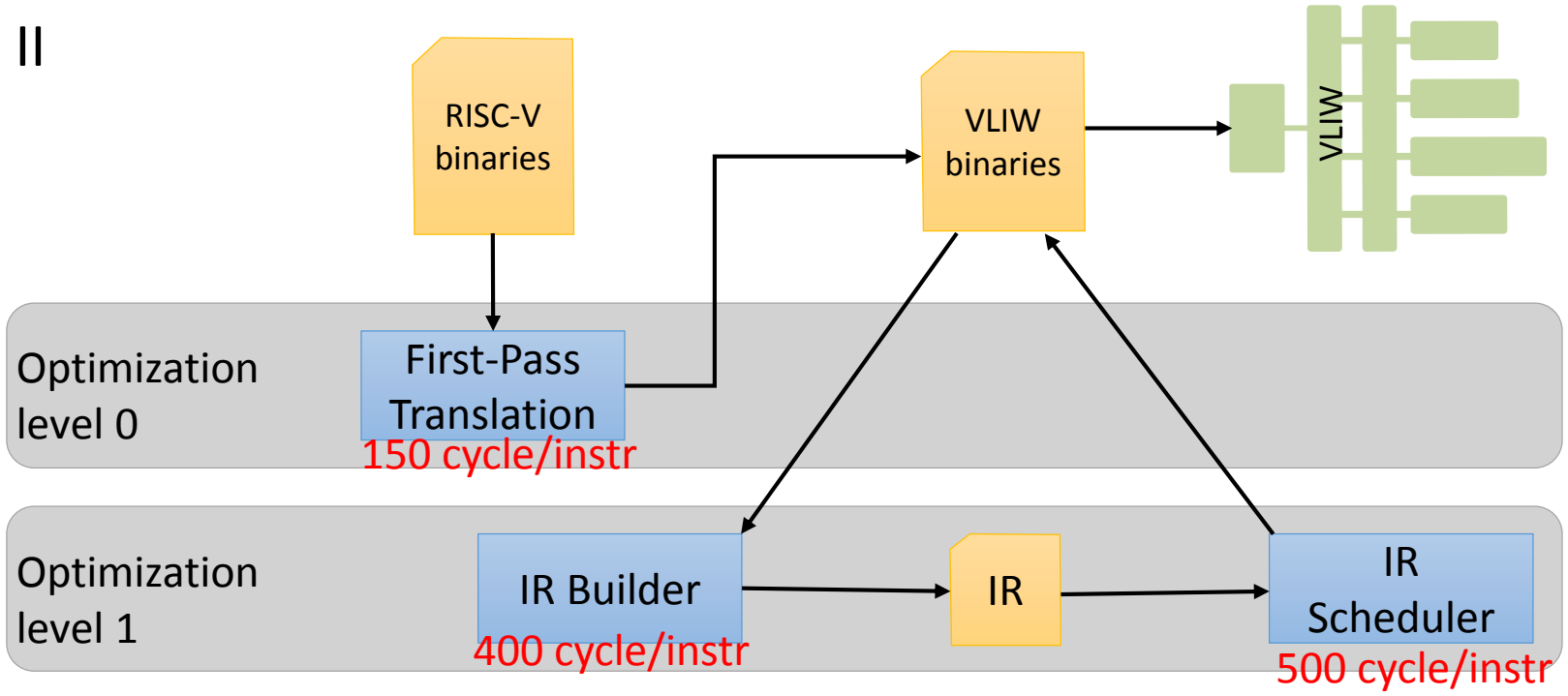
See paper for more details !

Outline

- Hybrid-DBT Platform
  - How does it work?
  - What does it cost?
  - Focus on optimization levels
- **Experimental Study**
  - **Impact on translation overhead**
  - **Impact on translation energy overhead**
  - **Impact on area utilization**
- Conclusion & Future work

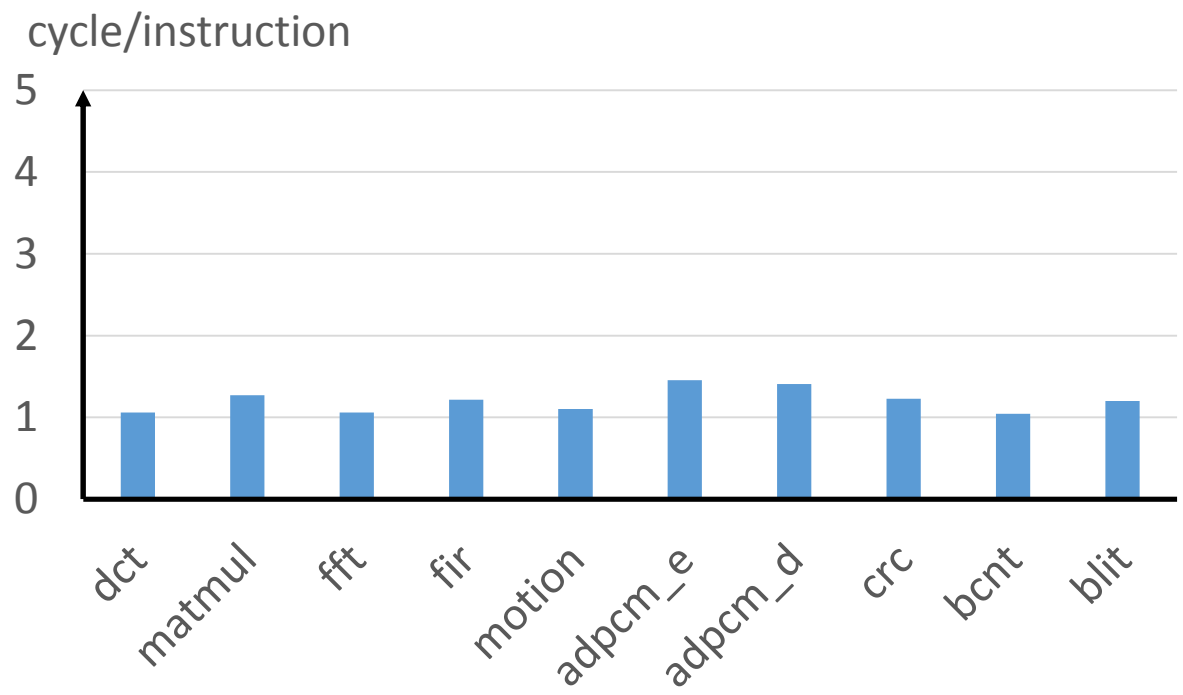
Impact on translation overhead

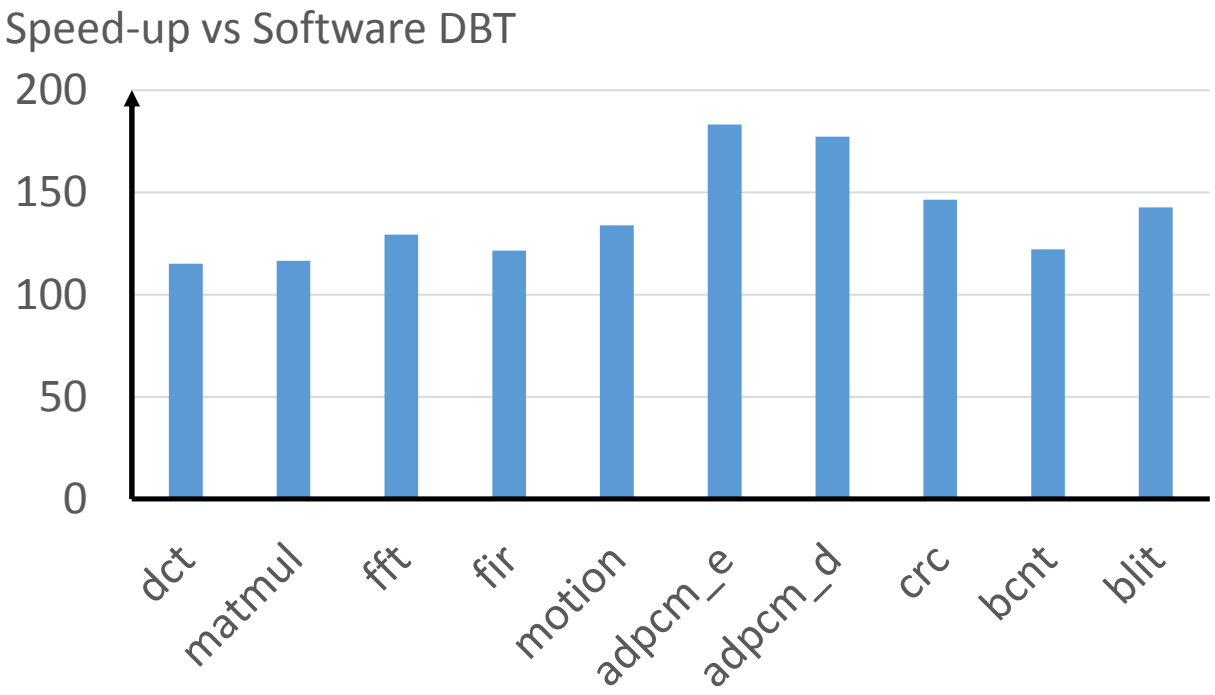
- VLIW baseline is executed with ST200simVLIW
- Fully functionnal Hybrid-DBT platform on FPGA
  - JIT processor: Nios II
  - Altera DE2-115



Impact on translation overhead

- Cost of **optimization level 0** using the hardware accelerator

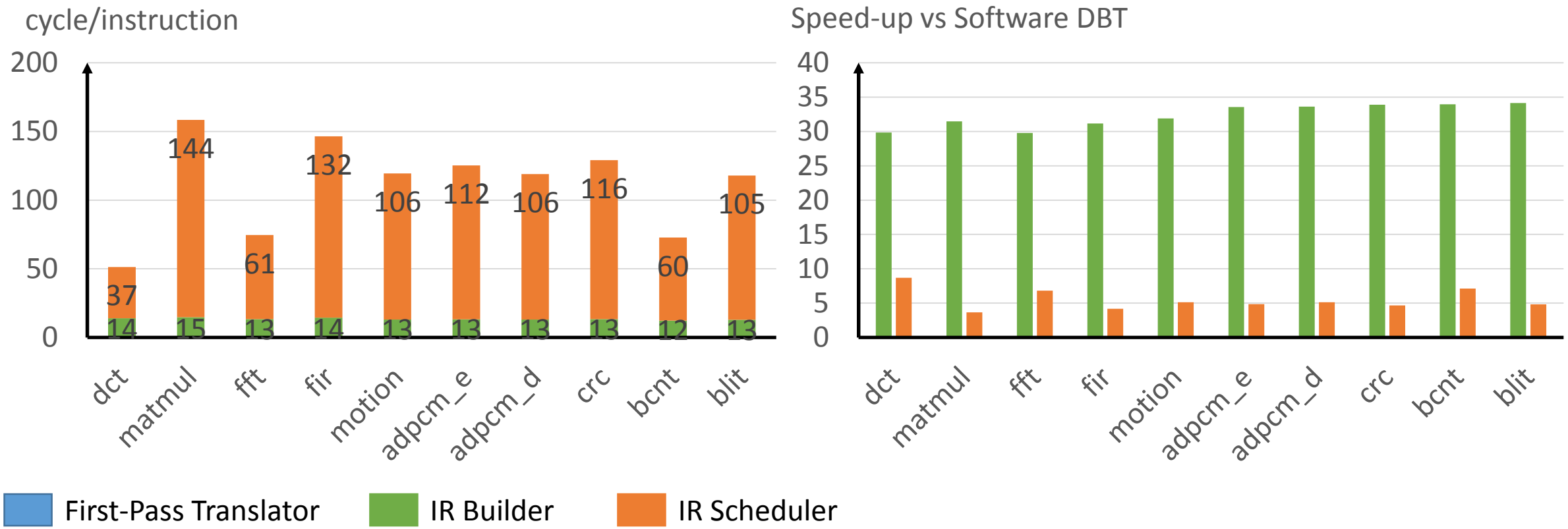




First-Pass Translator IR Builder IR Scheduler

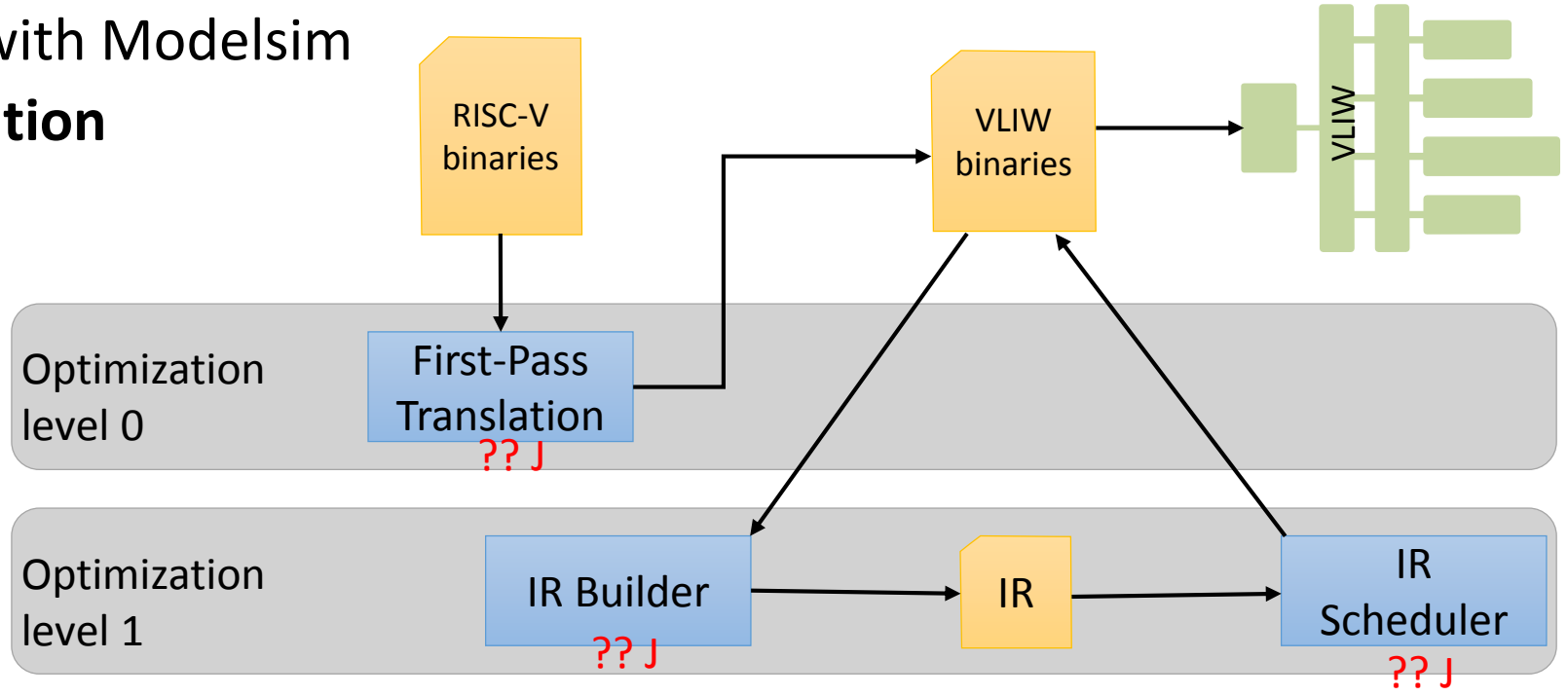
Impact on translation overhead

- Cost of **optimization level 1** using the hardware accelerator



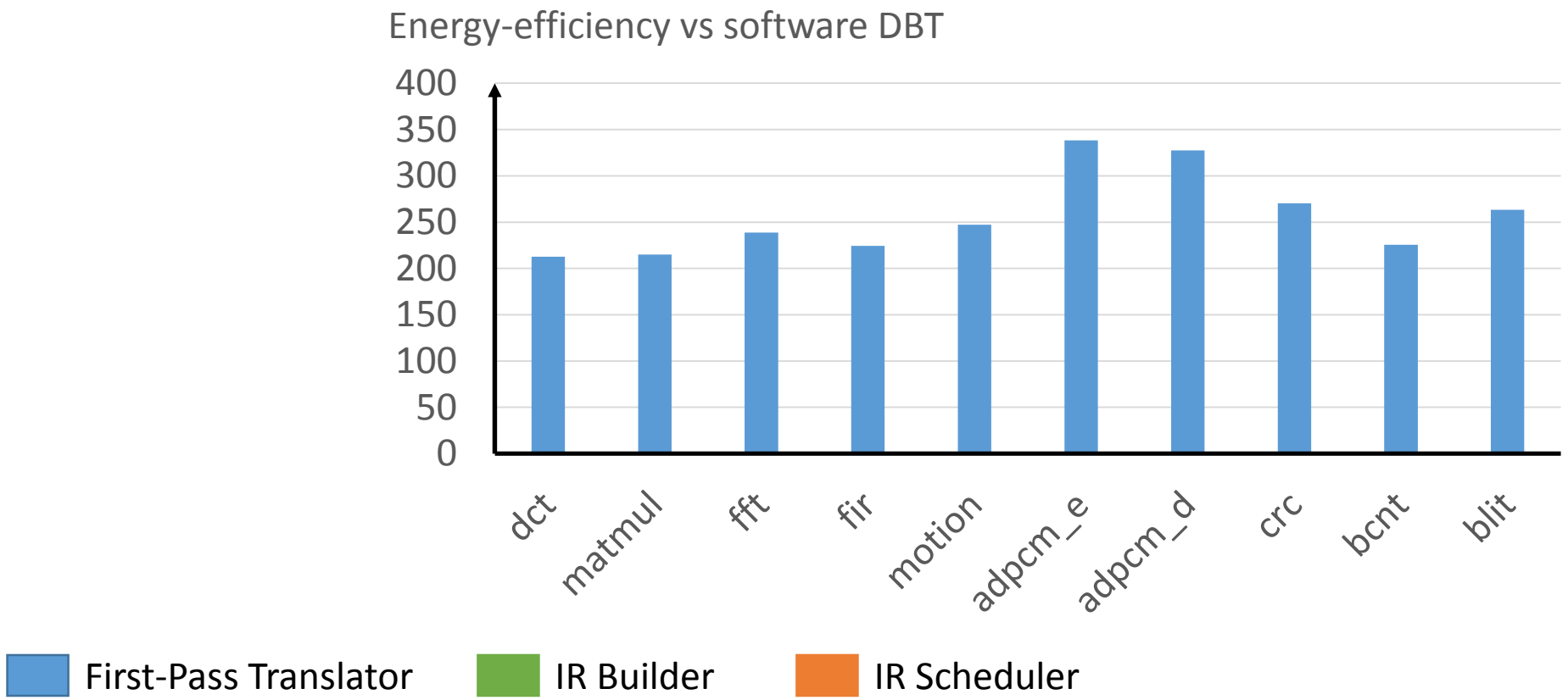
Impact on translation energy overhead

- Hybrid-DBT platform on ASIC:
  - Compiled with design compiler for **ASIC 65nm**
  - Design frequency: 250 MHz
  - **Gate-level simulation** with Modelsim
  - **Accurate power estimation**



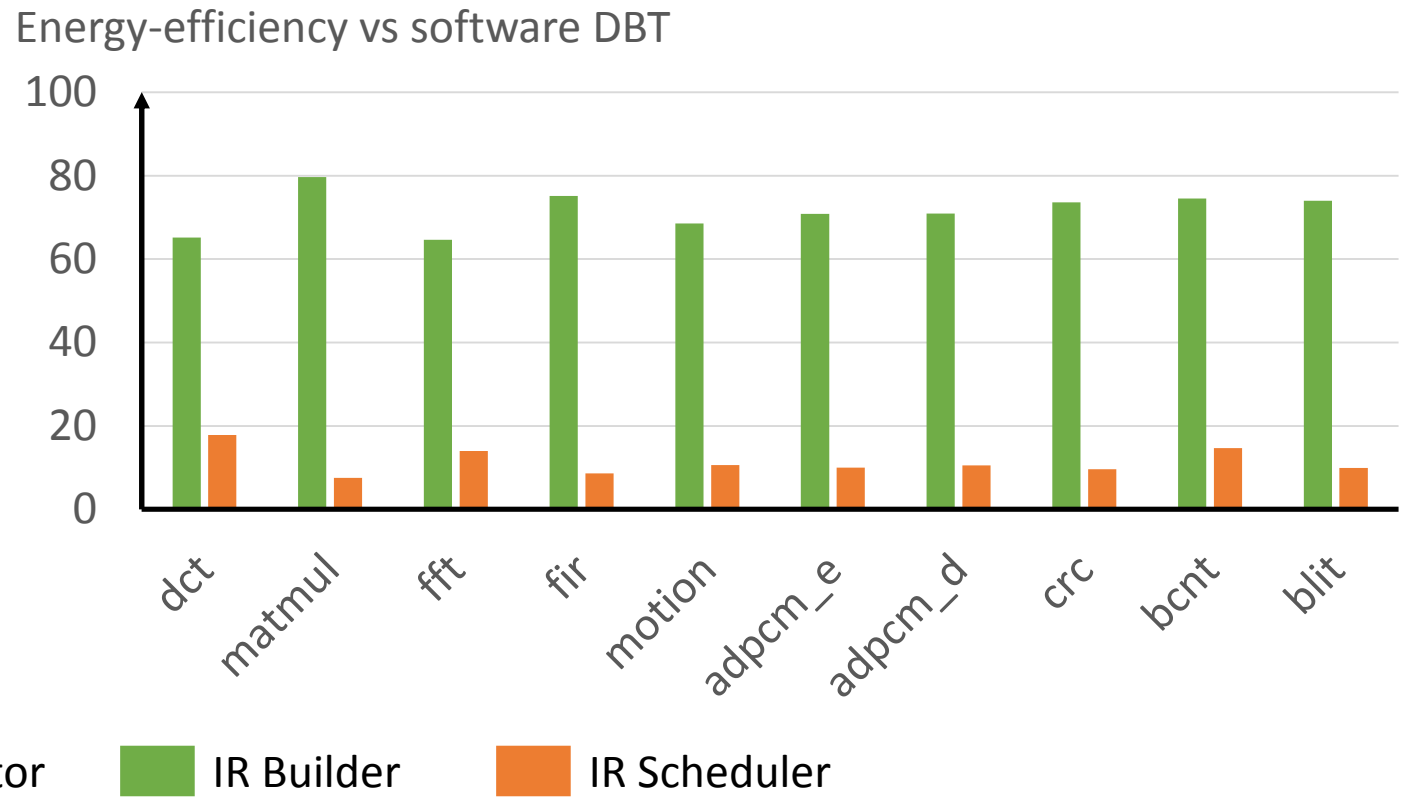
Impact on translation energy overhead

- Energy-efficiency improvement using the hardware accelerator



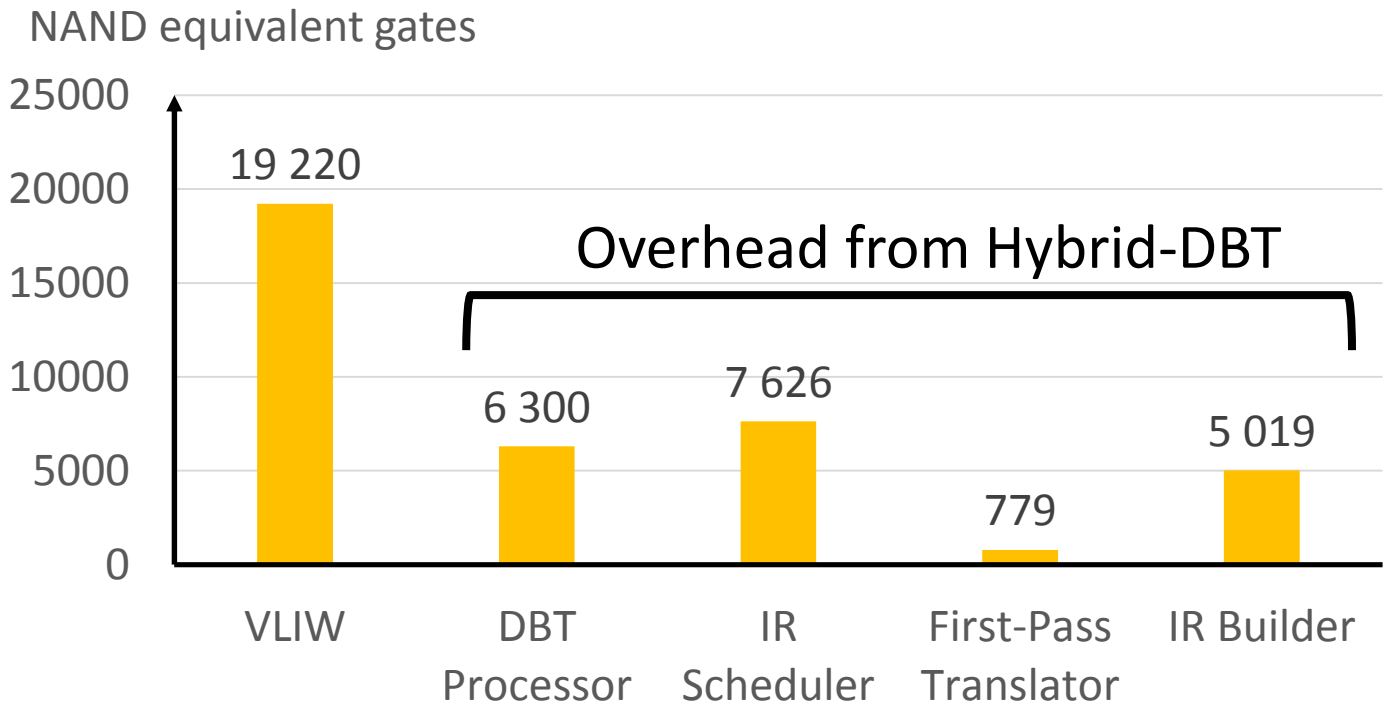
Impact on translation energy overhead

- Energy-efficiency improvement using the hardware accelerator



Impact on area/resource cost

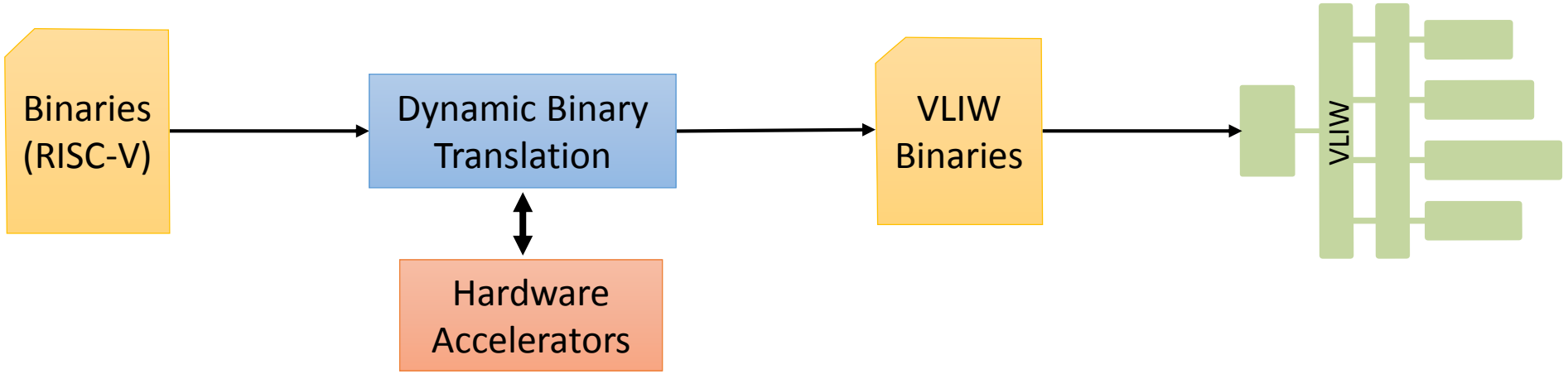
- Resource usage for all our platform components



Conclusion

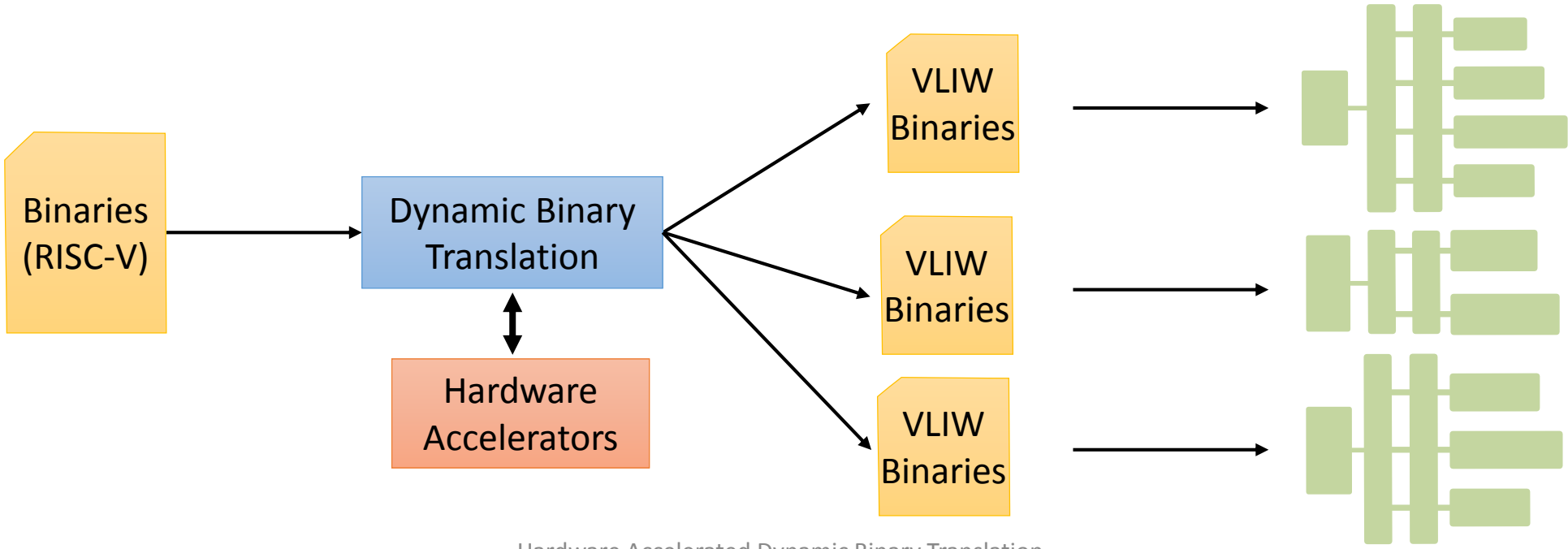
- Presentation of Hybrid-DBT framework
  - Hardware accelerated DBT
  - Open-source DBT framework RISC-V to VLIW
  - Tested FPGA prototype

- Sources are available on GitHub: https://github.com/srokicki/HybridDBT

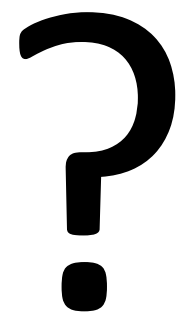


Future Work

- DBT to support hardware adaptability
- Exploring cost/impact of optimizations
- Comparison with existing RISC-V implementations (BOOM)



Questions



https://github.com/srokicki/HybridDBT