Institute of Microelectronic Systems

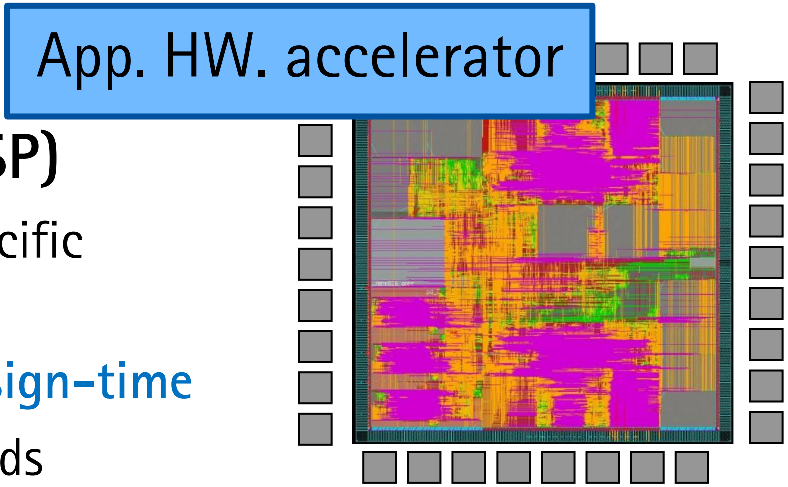
Leibniz Universität Hannover

rASIP: Reconfigurable Application Specific Processors

Challenges and Opportunities

Guillermo Payá-Vayá, Stephan Nolting, Florian Giesemann, and Holger Blume

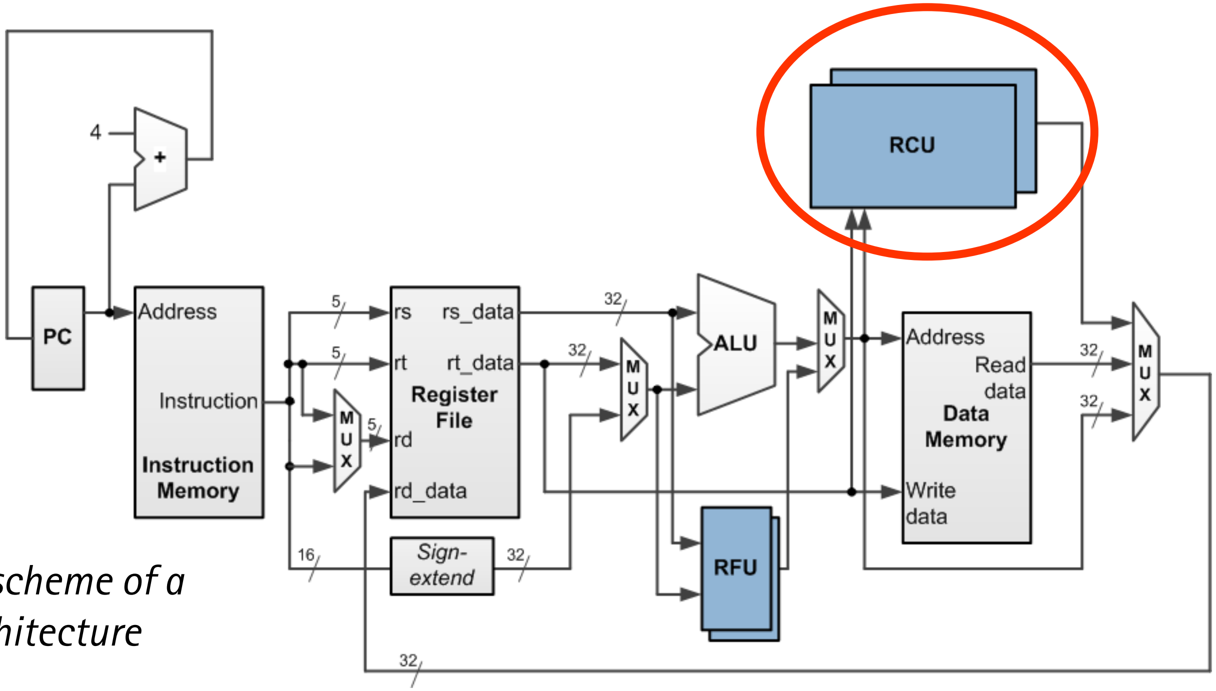
Motivation – *Why reconfigurable ASP?*



- Application Specific Processor (ASP)
  - Baseline processor optimized for a specific application or application set
  - Specialization is performed during design-time
  - Resulting processing efficiency depends on the applications (benchmarks) used during design-time
    - Increase reusability by implementing "general purpose" application hardware accelerator (e.g., counting leading zeros)
- Reconfigurable Application Specific Processor (rASP)
  - Baseline processor optimized for a specific application or application set
  - Specialization is performed during design-time and run-time
  - Resulting processing efficiency can be improved during run-time
    - Reusability compatible with more "specific" application hardware accelerator

A Reconfigurable MIPS (rMIPS) Processor (I)

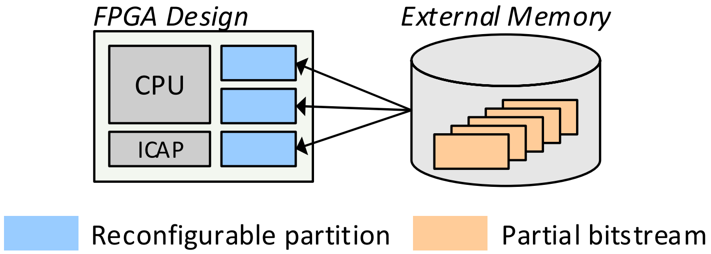
- 32-bit MIPS processor, 5 pipeline stages, C-compiler available (LLVM), I-/D-Cache
  - RFU (reconfigurable functional unit) for small tasks (custom instructions)
  - RCU (reconfigurable coprocessor unit) for more complex tasks (e.g. SINE)



Simplify scheme of a MIPS architecture

A Reconfigurable MIPS (rMIPS) Processor (II)

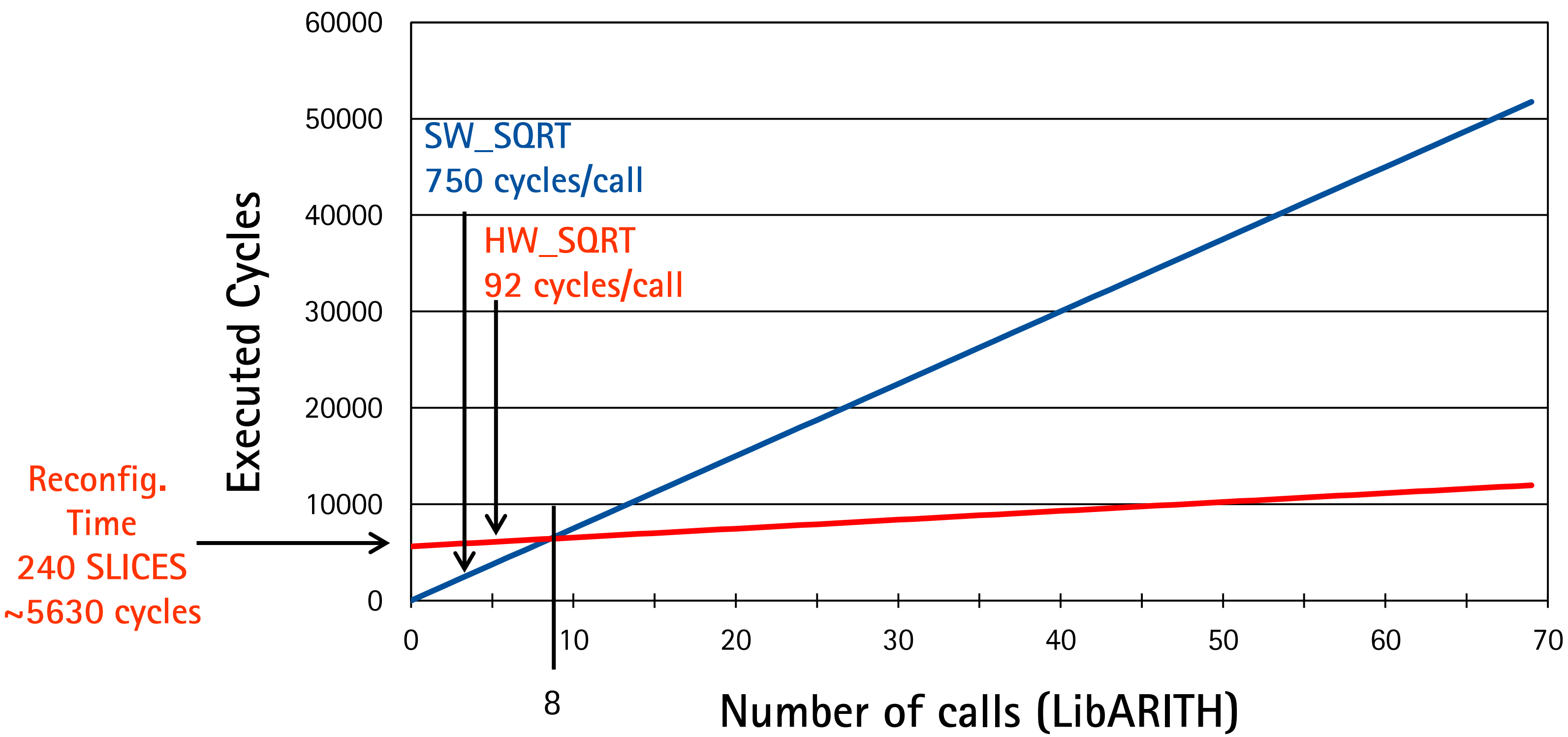
- Proof of concept on a Virtex-6 Xilinx FPGA
- Mathematical LibARITH library in software and hardware (RCU)
    - Sine and cosine (SINCOS)
    - Square root (SQRT)
    - Exponential function (EXP)
    - Natural logarithm (LN)
    - 3x3x6 Matrix multiplication (MM3x3x6)
    - 6x6x6 Matrix multiplication (MM6x6x6)



Function	#SW (Cycles)	#HW (Cycles)
SINCOS	232	90
SQRT	750	92
EXP	300	128
LN	321	82
MM3x3x6	1225	456
MM6x6x6	4288	906

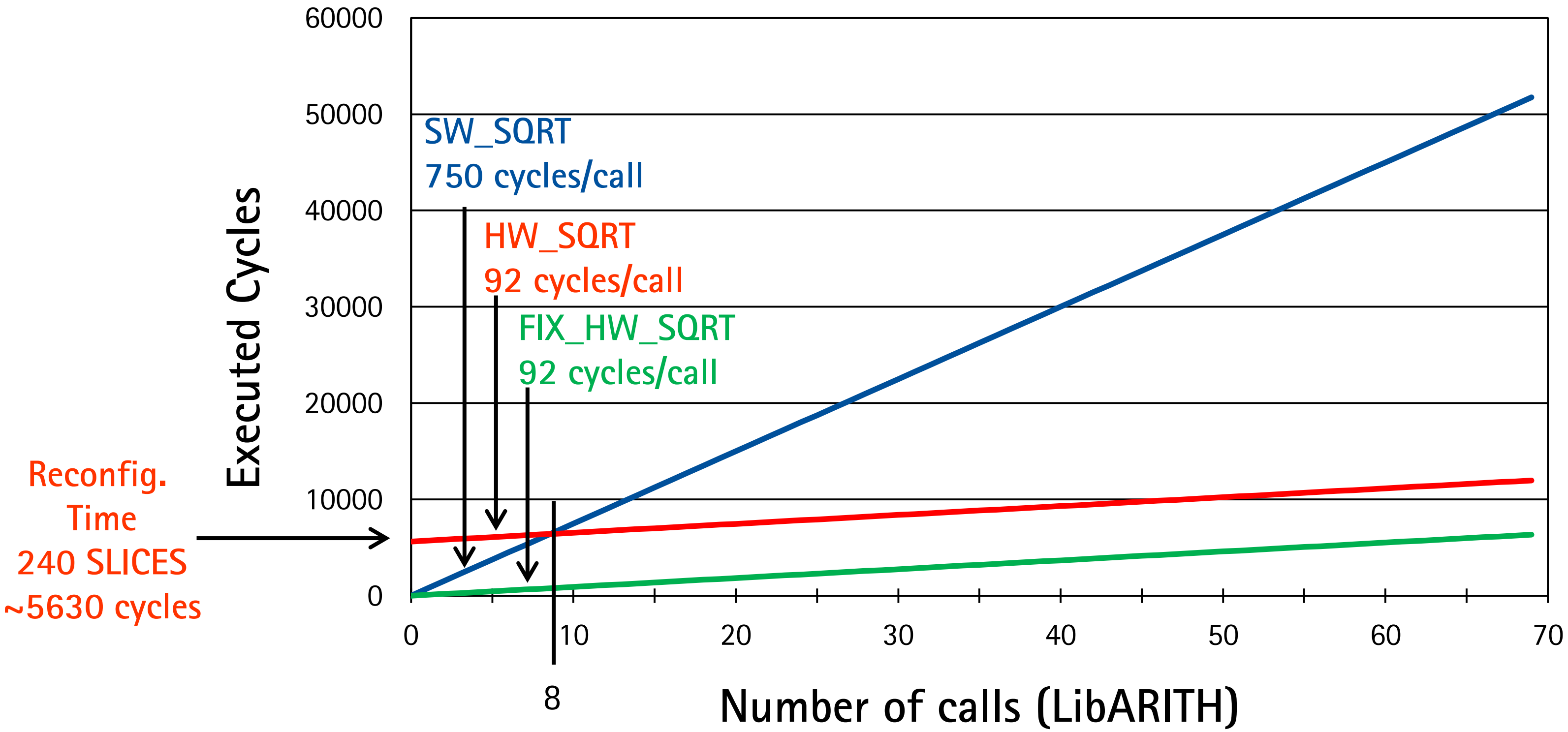
rMIPS: HW–SW LibARITH Library

Challenge: Time required during reconfiguration process



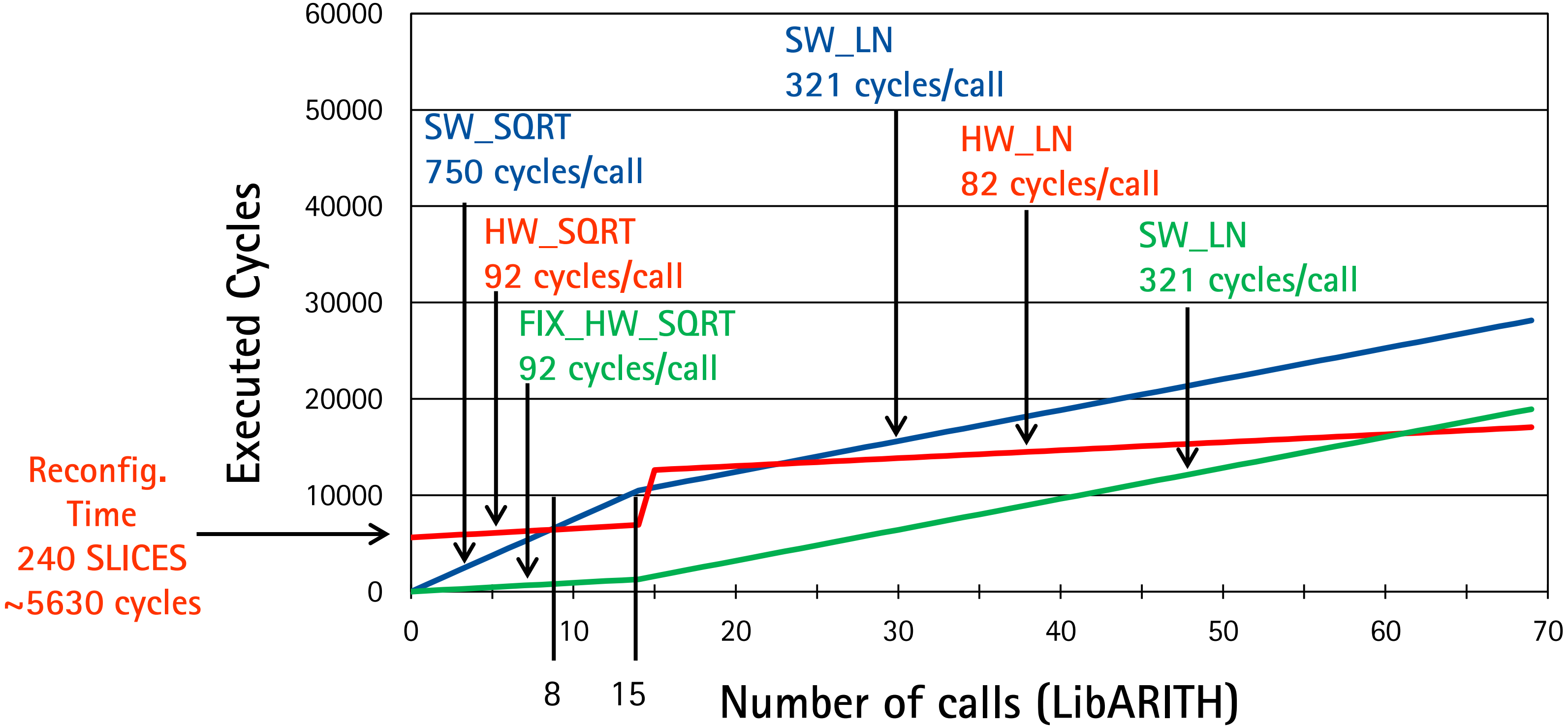
rMIPS: HW–SW LibARITH Library

Challenge: Time required during reconfiguration process



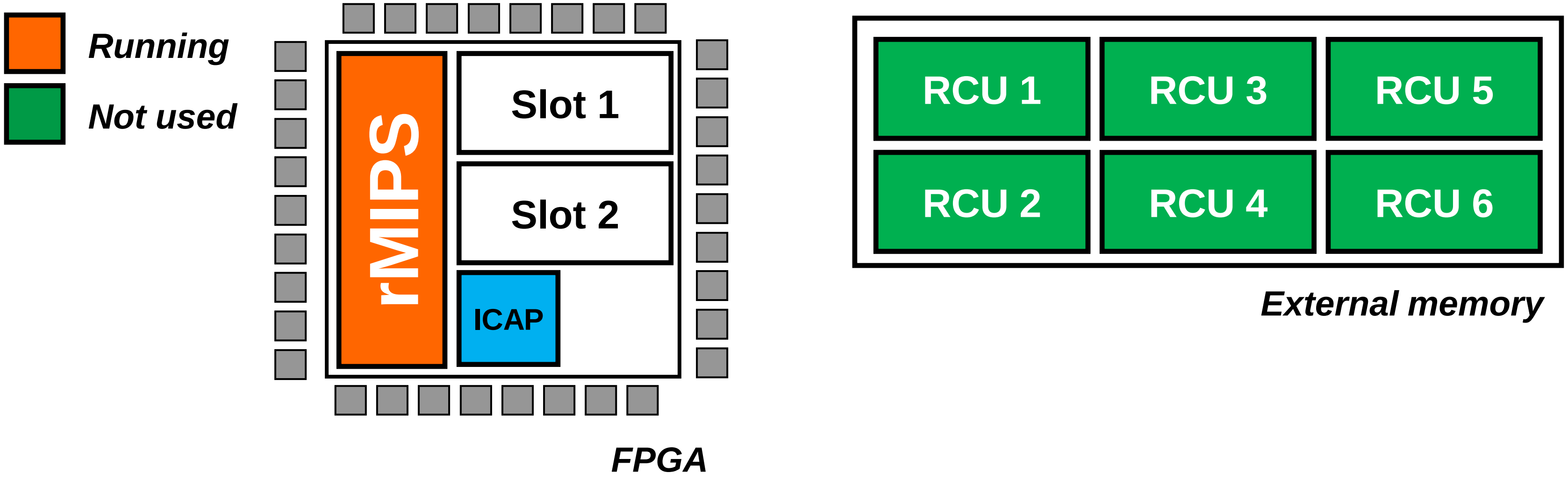
rMIPS: HW–SW LibARITH Library

Challenge: Time required during reconfiguration process



rMIPS: HW–SW LibARITH Library

Challenge: Time required during reconfiguration process



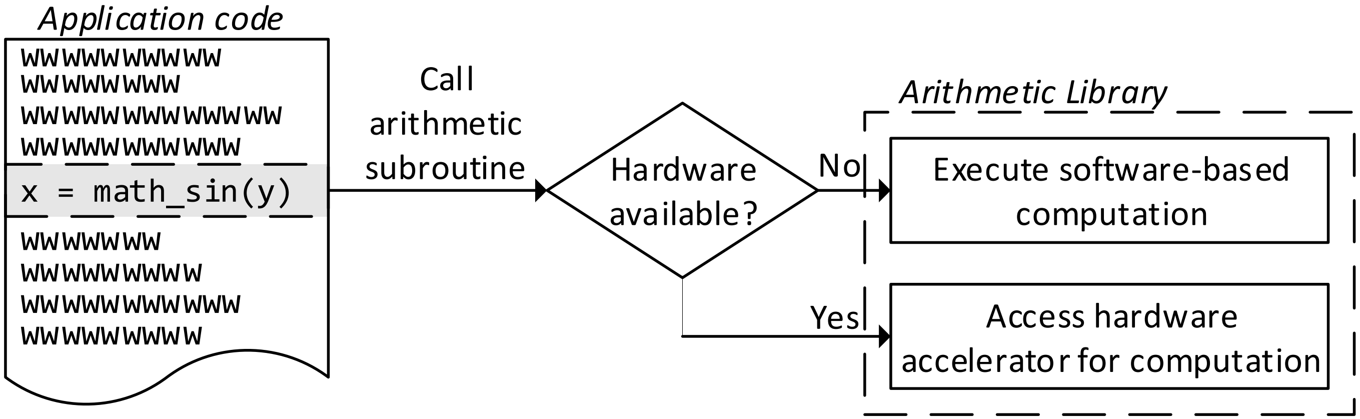
Opportunity: Unlimited number of "specific" app. HW. accelerators

rMIPs – Reconfiguration (RCU-Slots) Scheduling

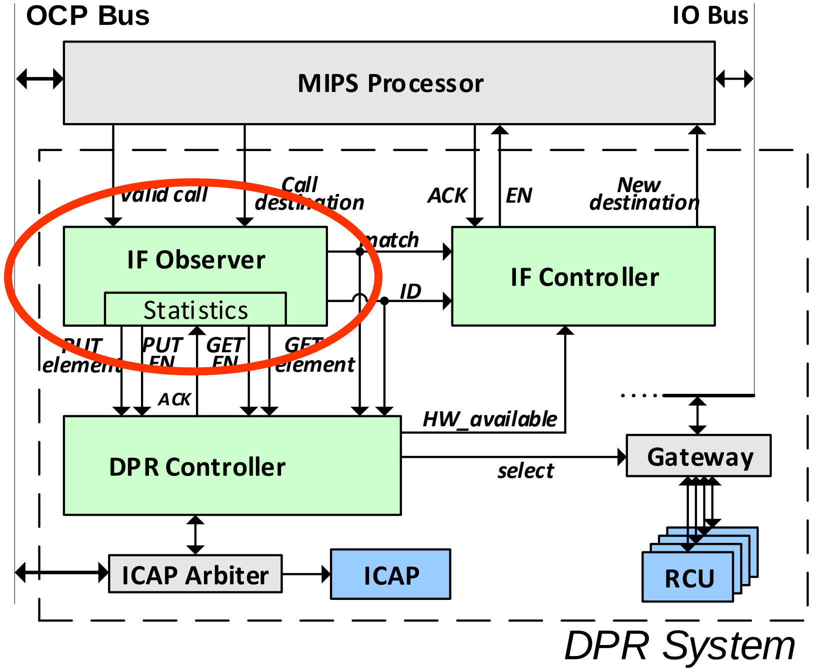
Challenge: Optimal reconfiguration scheduling could be not deterministic on design-time (e.g., profiling)

LibARITH provides pure SW-based and HW-accessing subroutines to compute a set of complex arithmetic operations (SINE, SQRT, ...)

Opportunity: Programmer does not need to think about explicitly using HW accelerators



rMIPS – Autonomous Reconfiguration (I)



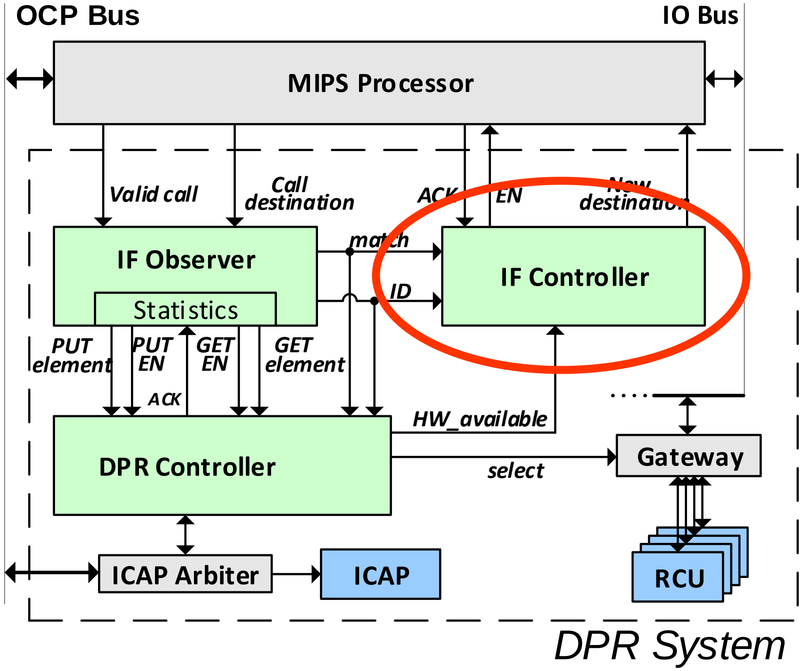
Access Counter (N)	SW Function Address
622	sw_sin: 0x00002030
451	sw_sqrt: 0x00004404
46	sw_exp: 0x00006080
0	sw_ln: 0x00010202

IF Observer

🟧 Managed by hardware 🟦 Programmable by user / application

- Dynamic Partial Reconfiguration (DPR) System
  - Instruction Fetch (IF) Observer

rMIPS – Autonomous Reconfiguration (II)



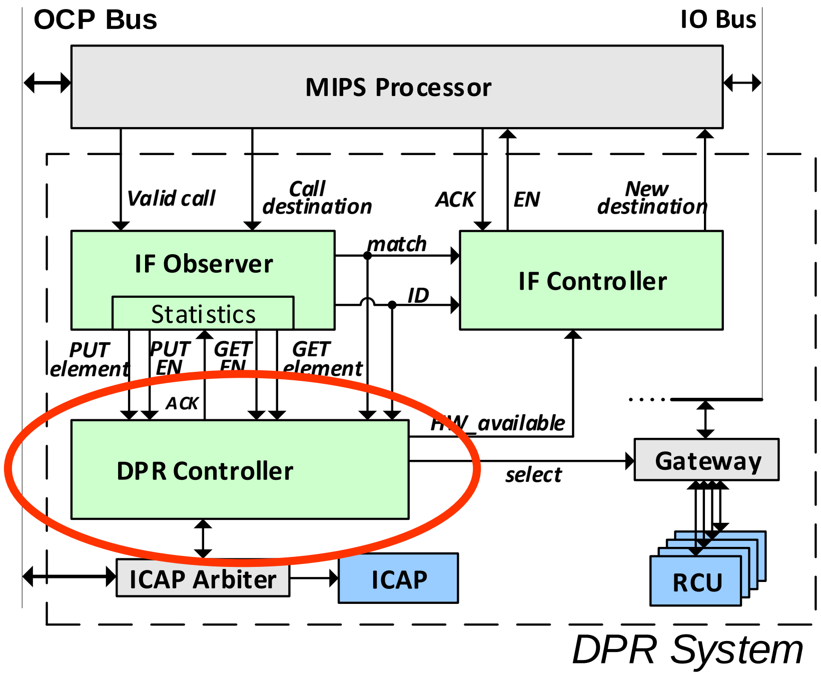
- Dynamic Partial Reconfiguration (DPR) System
  - Instruction Fetch (IF) Observer
  - Instruction Fetch (IF) Controller

Access Counter (N)	SW Function Address	HW Function Address	HW avail?	Corresp. RCU
622	sw_sin: 0x00002030	hw_sin: 0x00022008	true	Slot 2
451	sw_sqrt: 0x00004404	hw_sqrt: 0x00023400	true	Slot 1
46	sw_exp: 0x00006080	hw_exp: 0x00024E00	false	-
0	sw_ln: 0x00010202	hw_ln: 0x00024FF4	false	-

IF Observer — IF Controller — DPR Controller

🟧 Managed by hardware 🟦 Programmable by user / application

rMIPS – Autonomous Reconfiguration (III)



- Dynamic Partial Reconfiguration (DPR) System
  - Instruction Fetch (IF) Observer
  - Instruction Fetch (IF) Controller
  - DPR Controller
    - Reconfiguration Strategy

Access Counter (N)	SW Function Address	HW Function Address	Bitstream Address	HW avail?	Corresp. RCU
622	sw_sin: 0x00002030	hw_sin: 0x00022008	sin_rcu @ 0x00100000	true	Slot 2
451	sw_sqrt: 0x00004404	hw_sqrt: 0x00023400	sqrt_rcu @ 0x00110000	true	Slot 1
46	sw_exp: 0x00006080	hw_exp: 0x00024E00	exp_rcu @ 0x00120000	false	-
0	sw_ln: 0x00010202	hw_ln: 0x00024FF4	ln_rcu @ 0x00130000	false	-

IF Observer IF Controller DPR Controller

🟧 Managed by hardware 🟦 Programmable by user / application

rMIPS– Reconfigurable Strategy (I)

- Hardware replacement policies derived from computer cache systems
  - First-In-First-Out (FIFO)
  - Random (RND)
  - Least-recently used (LRU)
  - Least-frequently used (LFU)

Implement an on-demand concept:

An RCU is always loaded when its corresponding function is accessed.

Function	#SW (Cycles)	#HW (Cycles)
SINCOS	232	90
SQRT	750	92
EXP	300	128
LN	321	82
MM3x3x6	1225	456
MM6x6x6	4288	906

rMIPS– Reconfigurable Strategy (II)

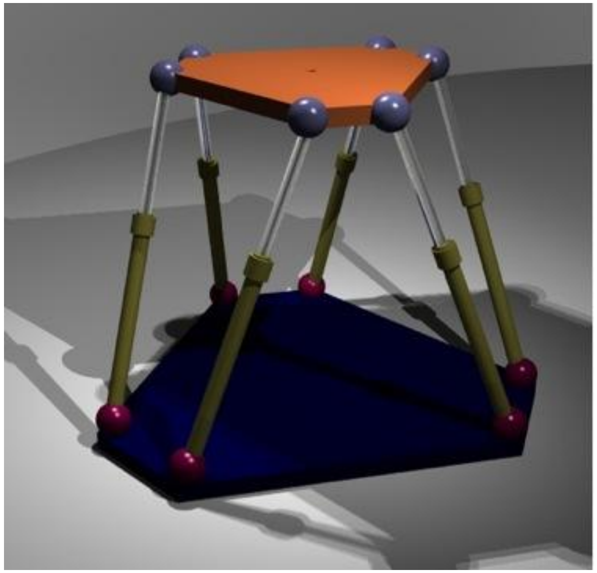
- Weighted Replacement Policy (WRP) with Hysteresis

  The called function corresponding to RCU_y will substitute RCU_x (already reconfigured and available) if $(N_x * W_{x,y}) + H < N_y$

Function	#SW (Cycles)	#HW (Cycles)
SINCOS	232	90
SQRT	750	92
EXP	300	128
LN	321	82
MM3x3x6	1225	456
MM6x6x6	4288	906

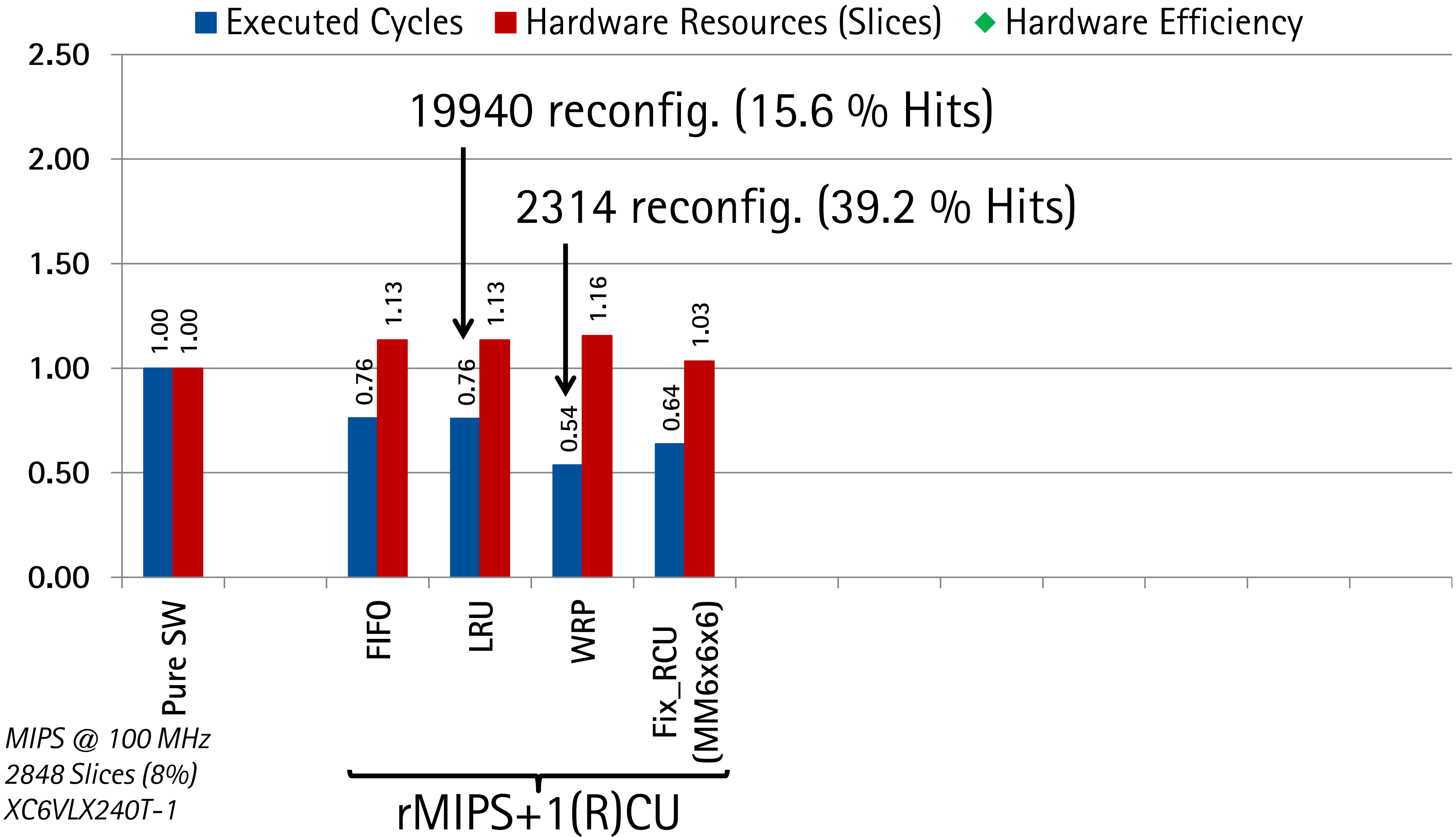
rMIPS – Evaluation with two benchmark

- An iterative synthetic benchmark consists of a large invocation sequence of LibARITH functions, featuring a recurring pattern separated by small random-distributed sections (e.g., simulate the behavior of data dependent applications)
  - All six functions are used
- A backtracking algorithm from the parallel robotic field. The position of a Stewart platform is estimated using the information provided by the sensor located in the motors.
  - SINCOS, SQRT, and MM3x3x6



Stewart platform

rMIPS – Evaluation with a iterative synthetic benchmark



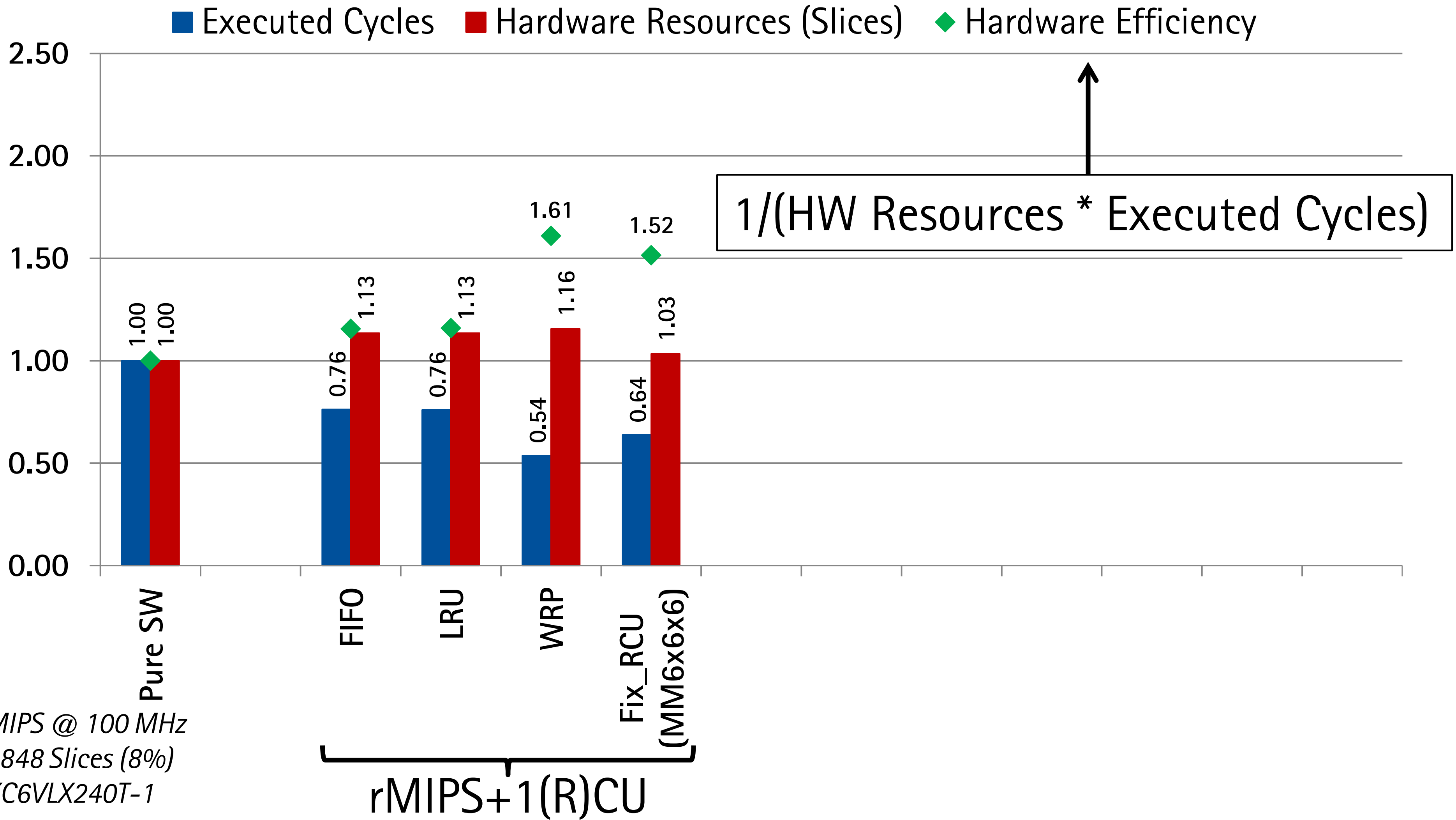
19940 reconfig. (15.6 % Hits)

2314 reconfig. (39.2 % Hits)

rMIPS+1(R)CU

MIPS @ 100 MHz
2848 Slices (8%)
XC6VLX240T-1

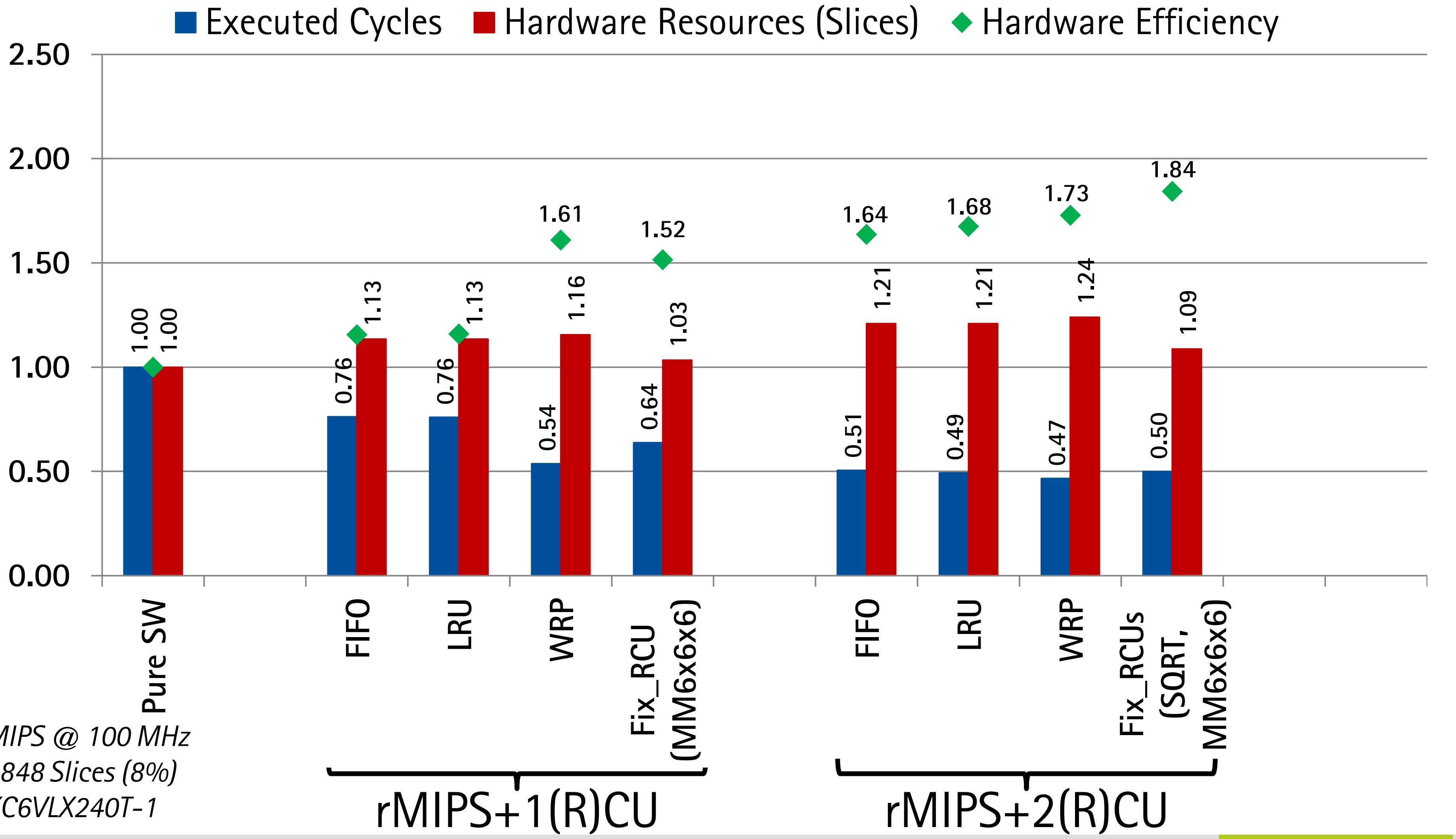
rMIPS – Evaluation with a iterative synthetic benchmark



1/(HW Resources * Executed Cycles)

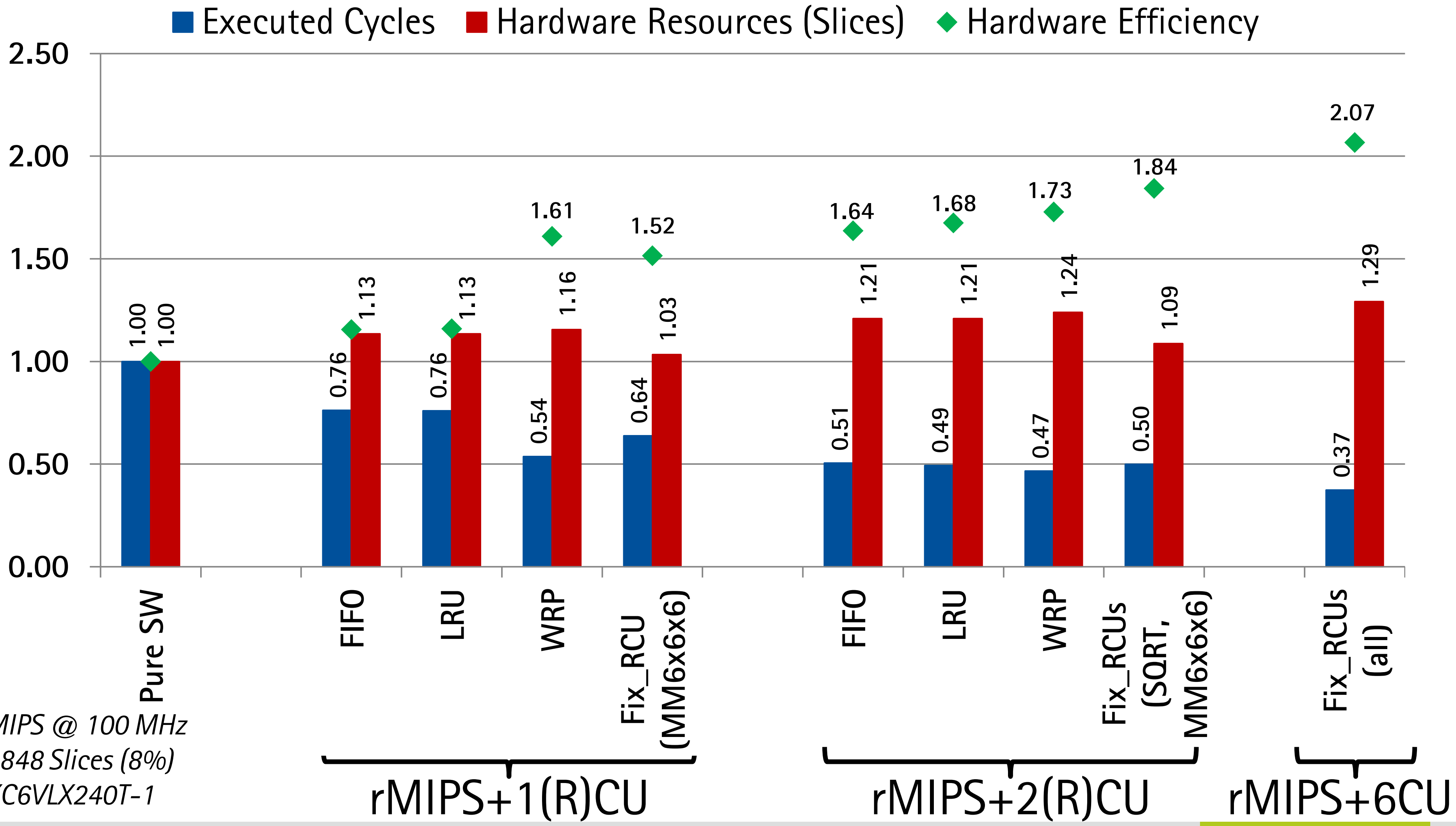
MIPS @ 100 MHz
2848 Slices (8%)
XC6VLX240T-1

rMIPS – Evaluation with a iterative synthetic benchmark



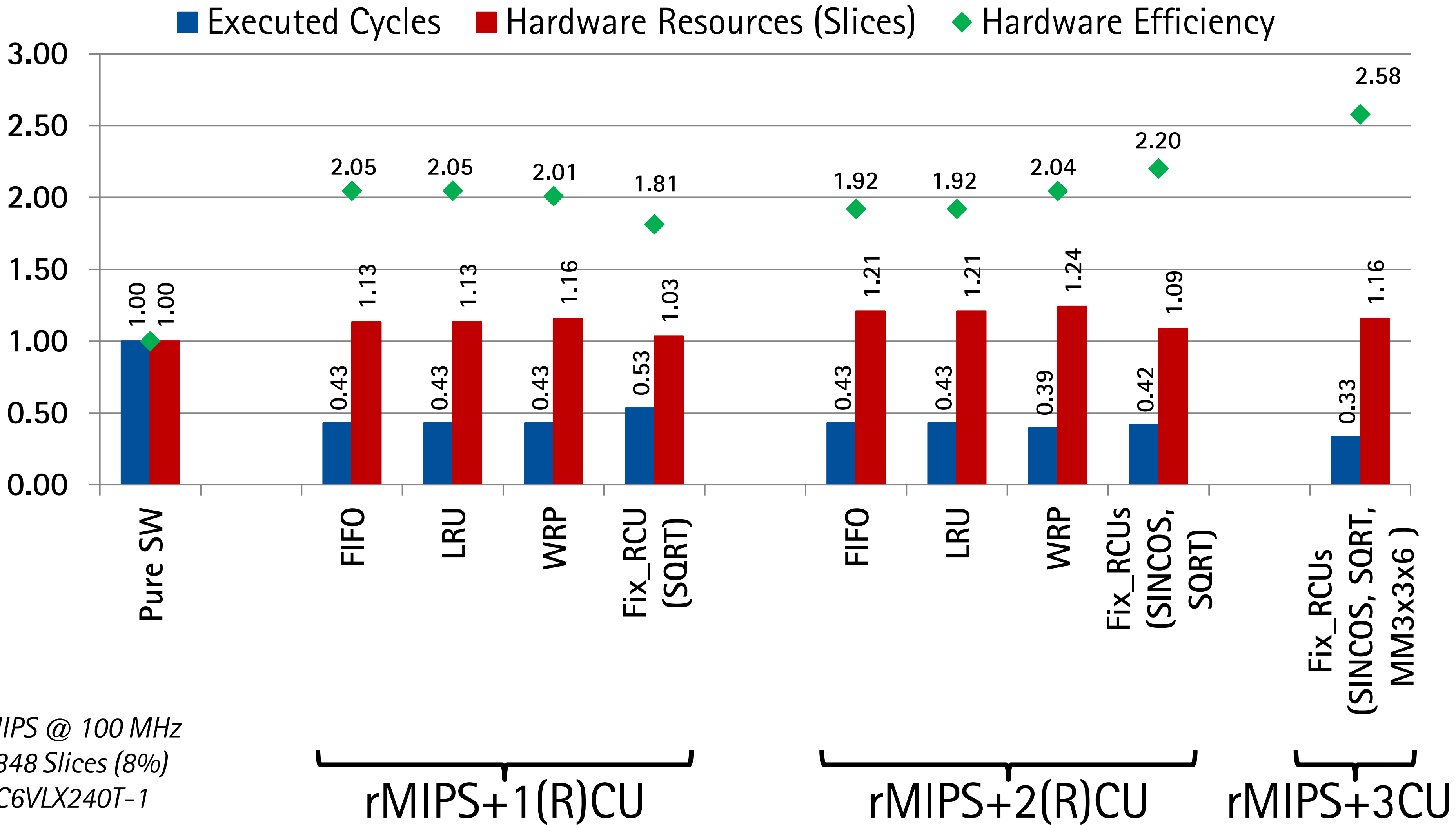
MIPS @ 100 MHz
2848 Slices (8%)
XC6VLX240T-1

rMIPS – Evaluation with a iterative synthetic benchmark



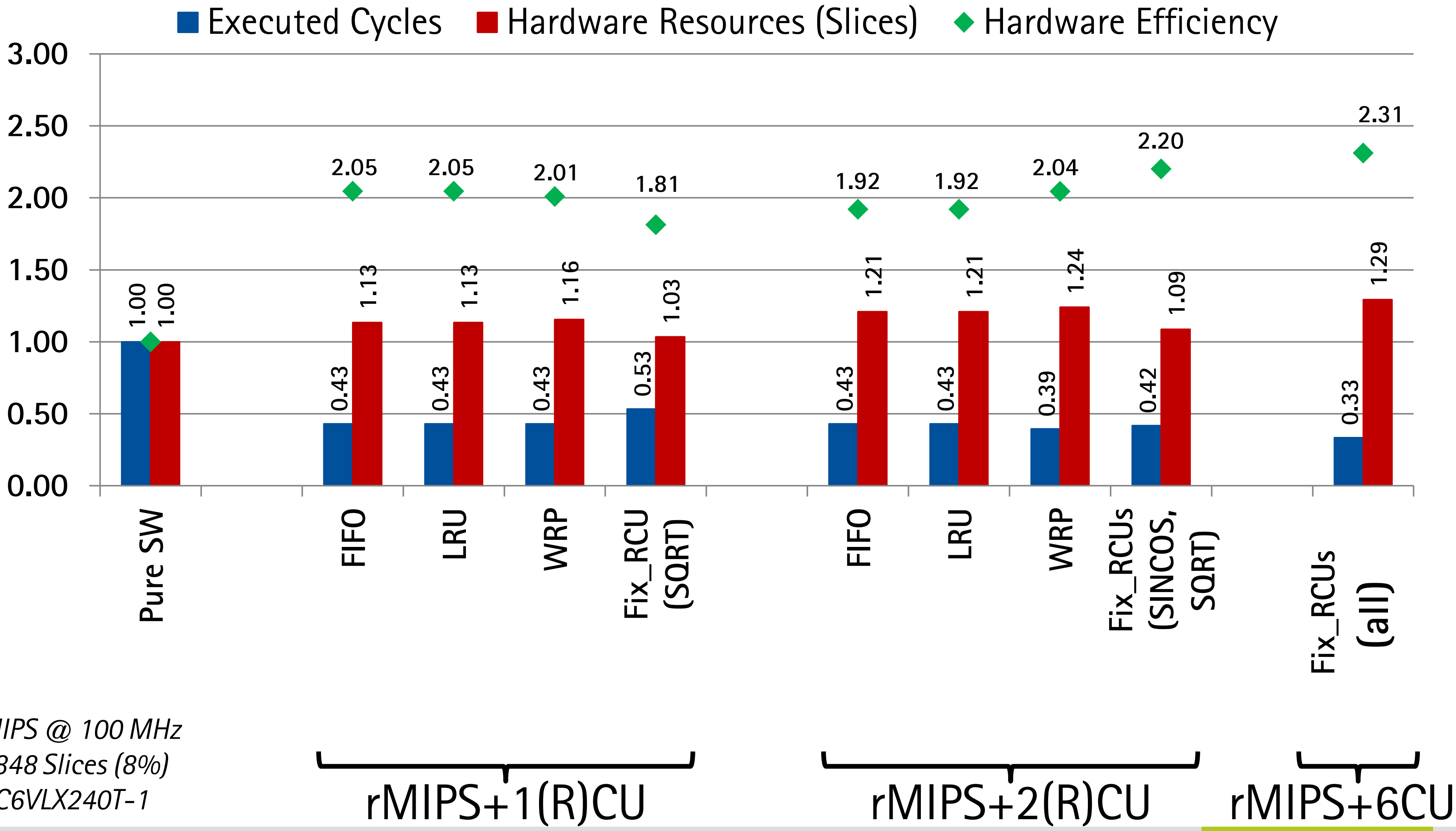
MIPS @ 100 MHz
2848 Slices (8%)
XC6VLX240T-1

rMIPS – Evaluation with a Parallel Robotics application



MIPS @ 100 MHz
2848 Slices (8%)
XC6VLX240T-1

rMIPS – Evaluation with a Parallel Robotics application



MIPS @ 100 MHz
2848 Slices (8%)
XC6VLX240T-1

Challenges and Opportunities

- **Opportunities:**
  - Unlimited number of RCUs
  - Programmer does not need to think about explicitly using HW accelerators
  - Evaluation with both benchmarks shows that:
    - rMIPS+1RCU is faster than a MIPS+1CU → 23%
      rMIPS+1RCU is more hardware efficient than a MIPS+1CU → 11%
    - rMIPS+2RCU is faster but less hardware efficient than a MIPS+2CU → 8%
- **Challenges:**
  - Predictive reconfiguration strategies are needed
  - Improve the LibARITH library
  - Reduce the time required to perform a reconfiguration
  - Improve the place-and-route of the RCUs on the reconfiguration partitions (since now less than 50% of hardware utilization)

Thank you for your attention!