# Examination of applicability of RISC-V security specifications to low-end processors

Fumio Arakawa† arakawa@silicon.u-tokyo.ac.jp
Akira Tsukamoto‡ akira.tsukamoto@aist.go.jp
Kuniyasu Suzaki§‡ k.suzaki@aist.go.jp
Makoto Ikeda† ikeda@silicon.u-tokyo.ac.jp

† *Graduate School of Engineering, The University of Tokyo,* Tokyo, Japan
‡ *Cyber Physical Security Research Center (CPSEC), National Institute of Advanced Industrial Science and Technology (AIST),* Tokyo, Japan
§ *Technology Research Association of Secure IoT Edge application based on RISC-V Open architecture (TRASIO),* Tokyo, Japan

***Abstract*—We evaluated the area overhead of security extensions on RISC-V that is an open instruction set architecture for general-purpose processors. Recently, the importance of security support is increasing, especially in the IoT era, the security of a large-scale system is affected by the vulnerability of a low-end processor in a terminal device of the system. However, it is important to reduce area and power of a low-end processor even by compromising on performance. The evaluated result shows the area of physical memory protection (PMP) function is comparable to that of the other parts of memory management unit, which greatly affects a low-end processor area. Further, the PMP area increases when virtual memory is enabled. The PMP area overhead reaches as much as 33.3% for a Mid RV32 Rocket tile with PMP=16.**

***Keywords—RISC-V, ISA, security, low-end, processor, area, PMP, TLB, MMU, Rocket, Chisel, Chipyard***

## I. Introduction

The importance of security support is increasing even in low-end processors with the recent trend of security enhancement. Especially in the IoT era, the security of a large-scale system is affected by the vulnerability of a low-end processor in a terminal device of the system. On the other hand, it is important to reduce area and power of a low-end processor even by compromising on performance. Under this background, we evaluated the area overhead of security extensions on RISC-V, which is gaining attention as an open instruction set architecture (ISA) for general-purpose processors.

## II. Overview of RISC-V

The RISC-V ISA [1, 2] was developed as a free and open ISA at UC Berkeley. It is currently maintained by RISC-V Foundation, whose member organizations have increased and exceed 325 [3]. It is gaining attention, and is expected to spread to actual products. 32- and 64-bit ISAs are defined, and specified by a parameter XLEN. The software of development and runtime environments is provided based on the ISA, and includes compilers, debuggers, simulators, boot loaders and OSes. Many RISC-V ISA cores have been developed along with fulfilling its ecosystem, and the official page lists 86 cores [4].

Berkeley Architecture Research released processor cores of Rocket [5, 6] and BOOM [7, 8] written in its original hardware construction language Chisel [9, 10]. It released a tool chain that automatically translates codes from Chisel to verilog, and Chipyard [11], a Rocket-based SoC construction environment.

## III. Security Extention of RISC-V

The security extension of RISC-V is now available as a draft version 1.12 [2]. This section is its overview of the related parts to this paper. Machine- (M-) mode is defined as the highest privilege level as well as conventional Supervisor- (S-) and User- (U-) modes for mainly OSes and applications, respectively. There are three implementation types assumed: a simple embedded system with the M-mode only, a secure embedded system with the M- and U-modes, and a system running an OS like Unix with all the M-, S- and U-modes. The mode switches to the M-mode after reset or unmasked interrupt.

Several control and status registers (CSRs) and privileged instructions are defined only for the M-mode. A physical memory attribute (PMA) checker manages a PMA of each area of the physical memory map of the entire system. The checker manages hardware-specific attributes with hardware, and the others with M-mode software, and informs the attributes to U- and S-mode software. Physical memory protection (PMP) for security enhancement is also realized in the M-mode. The details are explained in the next section.

Several CSRs and privileged instructions are defined for the S-mode, and can be used in the S- and M-modes. Page-based virtual memory (VM) is defined as an S-mode function, and the relevant CSR and instructions for the VM are defined. Three types of the VM, Sv32, Sv39, and Sv48, are defined, and handle 32, 39, and 48-bit addresses, respectively. Assuming hardware implementation, a page table walk (PTW) is defined to accelerate miss handling of a translation lookaside buffer (TLB).

### A. Physical Memory Protection (PMP)

Table I. Field Assignment of pmpcfg

| bit | name | description |
|---|---|---|
| 7 | L | Lock control (0: writable, 1: write is locked) |
| 6-5 | | reserved |
| 4-3 | A | Address-matching mode<br>0: OFF Null region (disabled)<br>1: TOR Top of range<br>2: NA4 Naturally Aligned (NA) 4-byte region<br>3: NAPOT NA Power-of-two region, ≥8 bytes |
| 2 | X | Execution control (0: not permitted, 1: permitted) |
| 1 | W | Write control (0: not permitted, 1: permitted) |
| 0 | R | Read control (0: not permitted, 1: permitted) |

| | |
|---|---|
| TOR: | An address A is matched when<br>(i-1)-th pmpaddr ≤ A < i-th pmpaddr for i-th entry (0 < i),<br>0 ≤ A < 0-th pmpaddr for 0-th entry. |
| NA4: | An NA address matches if it is in the 4-byte region of pmpaddr. |
| NAPOT: | When bit j of pmpaddr is the first '0' from the LSB, an NA address matches if it is in the $2^{j+3}$-byte region of the address made by clearing lower j bits of the pmpaddr to '0's. |

PMP function uses PMP CSRs to control the PMP of read, write, and execute for secure processing. The granularity of the control is platform-specific, but the standard supports as small as 4 bytes. The PMP is basically managed by M-mode software although a certain region privilege can be fixed by hardware.

Each PMP entry consists of a pair of 8-bit configuration CSR (pmpcfg) and XLEN-bit address CSR (pmpaddr), and the number of the entries is 16 at maximum. Each entry is accessible only in the M-mode, but an entry with L-bit set cannot be updated even in the M-mode, and is to be cleared by reset.

Table I shows the field assignment of each pmpcfg, which is packed into four CSRs for RV32 and two CSRs for RV64. The pmpaddr supports 4-byte granularity, its higher 10 bits are fixed to '0's for RV64, and the addressable range is 0 to $2^{34}$-4 in RV32 and $2^{56}$-4 in RV64. The smaller number entry has priority if multiple entries are matched. Then, a part of a region can be set to have another access privilege by this priority.

Table II. LIST OF PMP CASES

| R | W | X | S-/U-mode | M-mode | |
|---|---|---|---|---|---|
| | | | | L=1 | L=0 |
| 0 | 0 | 0 | – – – | | r w x |
| 0 | 0 | 1 | – – x | | r w x |
| 0 | 1 | 0 | reserved | | reserved |
| 0 | 1 | 1 | reserved | | reserved |
| 1 | 0 | 0 | r – – | | r w x |
| 1 | 0 | 1 | r – x | | r w x |
| 1 | 1 | 0 | r w – | | r w x |
| 1 | 1 | 1 | r w x | | r w x |

Table II lists PMP cases. The access privilege is checked with the R, W, X, and L of the matched highest priority entry and the security mode. The r, w, and x indicate read, write, and execute permissions, respectively, and '–' indicates no permission. An access fault exception occurs for the read, write, or execution if it is not permitted. The combination of R=0 and W=1 is not necessary, and reserved for future extension. Any access is permitted in the M-mode and no access is permitted in the S-/U-mode if no entry matches.

It is effective to skip changing access permission in the M-mode if the M-mode code is highly reliable. So, full access is permitted if L=0 in the M-mode. However, it is better to execute the M-mode code with minimum access privilege if it becomes bloated and less reliable. For this reason, an extension of the PMP specification is currently discussed.

### B. PMP implementation in Rocket Tile

Rocket tile is a Rocket core conforming to Tilelink [12, 13]. The Chipyard [11] supports five configurations of Big, Mid and Small RV64 tiles and Big and Tiny RV32 tiles, but other types are configurable by setting parameters.

Figure 1 shows PMP implementation in the Rocket tile. Each tile has modules of Rocket including the CSRs, Frontend, Data cache (DC), Tile I/F, and so on. The Big tile has also BTB (Branch Target Buffer) in the Frontend, and FPU. Further, the Big and Mid tiles have also memory management unit (MMU) consisting of TLBs in the DC and Frontend, and PTW, and the MMU manages Sv39 VM for RV64 and Sv32 VM for RV32.

The PMP is distributed and implemented in the CSRs, TLBs and PTW. Regardless of the configuration, the PMP of the Rocket tile in the Chipyard is 8 entries, which is half of the maximum, and the pmpaddr is 30 bits corresponding to a 32-bit physical address. Adding used 6 bits of the pmpcfg, one entry is 36 bits, and the total is 288 bits for the 8 entries.

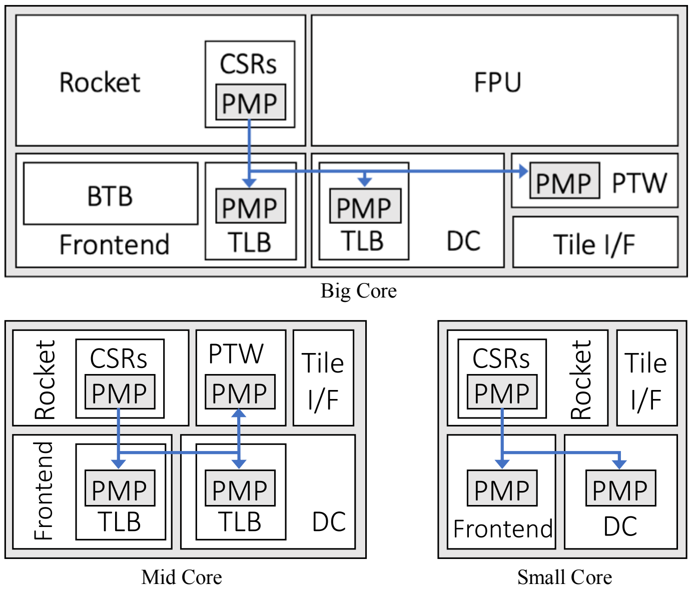

Fig. 1. PMP implementations in Rocket Tile

## IV. AREA EVALUATION OF PMP

Since a smaller core must be affected more by adding the PMP, we added Mid and Small RV32 tiles to the existing five types, and used totally six types excluding the Tiny tile, whose area is similar to the Small tile's. Then, we evaluated the three cases of PMP=0, 8 and 16 for each type.

First, we prepared totally 18 evaluating configurations by modifying two "Configs.scala"s of subsystem and system of rocket-chip in generators of the Chipyard, and created 18 RTL verilog descriptions of the tiles with the verilator construction environment of the Chipyard.

Next, we synthesized a gate-level verilog from each RTL verilog using Synopsis Design Compiler and Renesas 65nm SOTB library, with setting the Rocket tile as the top module, without hierarchical structure flattening. Then, we got the total cell area (not including RAM area; hereinafter simply called area) of the gate-level verilog of each tile. Further, we obtained the module areas from each gate-level verilog.

Table III shows the areas and increments of the modules of the 6 types, which are the relative values when the Small RV32 Rocket tile with PMP=0 is 100, and Table IV shows the area and increment ratios of the modules of each type, which is the percentage to the area of each Rocket tile with PMP=0. A column of PMP=0 shows the areas of modules excluding the PMP, and columns of PMP=8 and 16 show the increments by adding the PMP. The CSR F/F increments correspond to the 288-bit F/Fs, combinatorial logic increments are distributed to the CSRs, TLBs, and PTW, and the total increments are shown as Rocket tile increments.

Figure 2 is the graph of Table III for PMP=16. The increments are placed to the left side to form the total increment for the PMP addition of each type. From Table III, the increments by the PMP are proportional to the number of entries and increase when the VM is enabled, but other parameters do not affect. Therefore, it is confirmed that a smaller core is affected more by adding the PMP. Further, even a larger Mid tile with the VM is affected more than a smaller Small tile without it.

Table III. MODULE AREAS OF EACH CONFIGURATION (Relative area when Small RV32 tile with PMP=0 is 100)

| | RV64 | | | | | | | | | RV32 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Big | | | Mid | | | Small | | | Big | | | Mid | | | Small | | |
| PMP | 0 | 8 | 16 | 0 | 8 | 16 | 0 | 8 | 16 | 0 | 8 | 16 | 0 | 8 | 16 | 0 | 8 | 16 |
| RocketTile | 550.9 | 22.2 | 44.2 | 190.0 | 21.7 | 43.7 | 147.5 | 15.8 | 32.6 | 321.3 | 22.2 | 43.6 | 129.5 | 21.4 | 43.1 | 100.0 | 14.4 | 29.7 |
| Rocket | 132.5 | | | 111.3 | | | 103.1 | | | 75.0 | | | 69.6 | | | 64.9 | | |
| CSRs | 20.1 | 8.7 | 17.6 | 19.7 | 8.8 | 17.4 | 12.8 | 6.8 | 14.0 | 16.3 | 9.1 | 17.8 | 16.2 | 8.8 | 17.6 | 11.0 | 6.1 | 13.0 |
| F/F | 8.8 | 3.5 | 6.9 | 8.7 | 3.5 | 6.9 | 5.4 | 3.5 | 6.9 | 7.0 | 3.5 | 6.9 | 6.9 | 3.5 | 6.9 | 4.5 | 3.5 | 6.9 |
| Frontend | 111.2 | | | 22.9 | | | 12.2 | | | 102.3 | | | 20.2 | | | 11.9 | | |
| BTB | 51.0 | | | | | | | | | 47.7 | | | | | | | | |
| TLB | 31.9 | 4.5 | 9.1 | 9.6 | 4.2 | 8.4 | | 4.2 | 8.5 | 29.3 | 4.5 | 9.2 | 9.4 | 4.2 | 8.9 | | 4.2 | 8.4 |
| DC | 56.3 | | | 28.4 | | | 17.3 | | | 48.4 | | | 21.8 | | | 12.5 | | |
| TLB | 36.1 | 5.3 | 10.3 | 10.6 | 5.1 | 10.4 | | 4.7 | 10.1 | 33.6 | 4.6 | 9.0 | 8.3 | 4.3 | 9.1 | | 4.0 | 8.3 |
| PTW | 12.1 | 3.7 | 7.2 | 12.1 | 3.6 | 7.5 | | | | 6.7 | 4.1 | 7.5 | 6.7 | 4.1 | 7.6 | | | |
| FPU | 223.3 | | | | | | | | | 77.7 | | | | | | | | |

Table IV. MODULE AREA RATIOS OF EACH CONFIGURATION (Percentage to each Rocket tile area with PMP=0)

| | RV64 | | | | | | | | | RV32 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Big | | | Mid | | | Small | | | Big | | | Mid | | | Small | | |
| PMP | 0 | 8 | 16 | 0 | 8 | 16 | 0 | 8 | 16 | 0 | 8 | 16 | 0 | 8 | 16 | 0 | 8 | 16 |
| RocketTile | 100.0 | 4.0 | 8.0 | 100.0 | 11.4 | 23.0 | 100.0 | 10.7 | 22.1 | 100.0 | 6.9 | 13.6 | 100.0 | 16.6 | 33.3 | 100.0 | 14.4 | 29.7 |
| Rocket | 24.1 | | | 58.6 | | | 69.9 | | | 23.3 | | | 53.8 | | | 64.9 | | |
| CSRs | 3.7 | 1.6 | 3.2 | 10.4 | 4.6 | 9.2 | 8.7 | 4.6 | 9.5 | 5.1 | 2.8 | 5.5 | 12.5 | 6.8 | 13.6 | 11.0 | 6.1 | 13.0 |
| F/F | 1.6 | 0.6 | 1.3 | 4.6 | 1.8 | 3.6 | 3.6 | 2.3 | 4.7 | 2.2 | 1.1 | 2.1 | 5.3 | 2.7 | 5.3 | 4.5 | 3.5 | 6.9 |
| Frontend | 20.2 | | | 12.0 | | | 8.3 | | | 31.9 | | | 15.6 | | | 11.9 | | |
| BTB | 9.3 | | | | | | | | | 14.8 | | | | | | | | |
| TLB | 5.8 | 0.8 | 1.6 | 5.1 | 2.2 | 4.4 | | 2.8 | 5.7 | 9.1 | 1.4 | 2.9 | 7.2 | 3.2 | 6.9 | | 4.2 | 8.4 |
| DC | 10.2 | | | 14.9 | | | 11.7 | | | 15.1 | | | 16.9 | | | 12.5 | | |
| TLB | 6.6 | 1.0 | 1.9 | 5.6 | 2.7 | 5.5 | | 3.2 | 6.9 | 10.4 | 1.4 | 2.8 | 6.4 | 3.3 | 7.0 | | 4.0 | 8.3 |
| PTW | 2.2 | 0.7 | 1.3 | 6.3 | 1.9 | 3.9 | | | | 2.1 | 1.3 | 2.3 | 5.2 | 3.2 | 5.8 | | | |
| FPU | 40.5 | | | | | | | | | 24.2 | | | | | | | | |

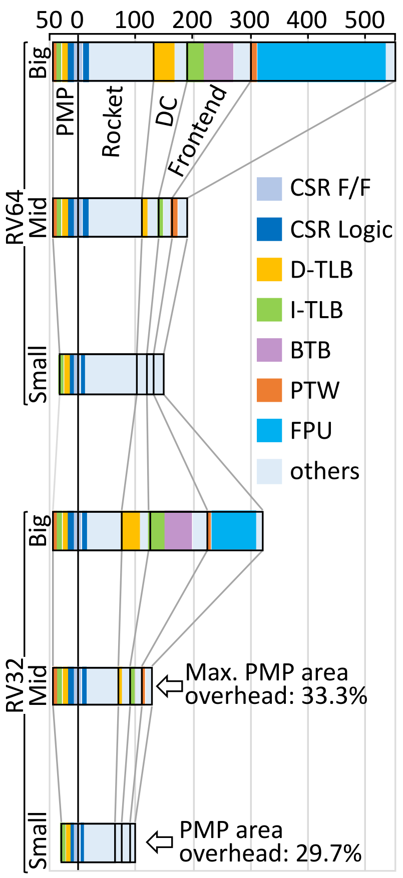

Fig. 2. Graph of Table III for PMP=16

From Table IV, the area increase ratios are 14.4 and 29.7% for the smallest Small RV32 tile with PMP=8 and 16, respectively. However, the ratios are 16.6 and 33.3% for the Mid RV32 tile with PMP=8 and 16, respectively. The increase is the largest even it is not the smallest tile because its VM is enabled.

## V. COMPARISON OF PMP AND TLB

### A. *Difference between PMP and TLB*

Here, the TLB is for the VM excluding the PMP function. Both the PMP and TLB are searched on each memory access, and the information of the selected entry is used. However, there are various differences affecting the area. The important difference is that the mapping is overlapped in the PMP and exclusive in the TLB. With the overlapped mapping, all the entries are to be searched and the highest priority entry is selected from the matched entries. Then all the entries must always be in the core. On the other hand, the TLB is a buffer that holds a part of a large page table with the exclusive mapping, and a hit entry is always the matching entry. Any number of the TLB entries is selectable with the exclusive mapping.

The next important difference is the region size variations. The variations of the TLB are 4 KiB and 4 GiB for Sv32, 4 KiB, 2 MiB, and 1 GiB for Sv39, and the Sv39's plus 512 GiB for Sv48, whereas the size for each PMP entry is freely from 4 bytes to the entire address space. Each entry must have an additional 10-bit comparator and a bitwise mask to realize the region of the 4-byte and the arbitrary size, respectively.

Since the PMP is checked at the PTW in addition to ordinary memory access, check logics are to be at three places: instruction and data TLBs, and the PTW, but no TLB is necessary for the PTW. Further, the number of the PMP CSRs is at most 18 for RV64 and 20 for RV32. In contrast, a page table is placed in memory, so VM CSR is only the supervisor address translation and protection (satp) register.

On the other hand, an increasing factor of the area of the TLB is the number of bits to be retained and output. The PMP outputs access permission. In addition, the TLB outputs a physical address and page fault exception information. Another factor is the difference between the number of logical and physical address bits since the TLB is searched with a logical address.

### B. *Area Comparison of PMP and TLB*

Table V shows the areas of the PMP, TLBs and PTW. A PMP area is the Rocket tile's area increment by adding the PMP. The area for PMP=16 is about 44 and 30 when the VM is enabled and disabled, respectively, and those are about half for PMP=8.

On the other hand, the TLB areas are 68.0 for RV64 and 62.8 for RV32, when instruction and data TLBs are 32 entries each and total 64. Those are 20.2 and 17.6 when TLBs are 4 entries each and total 8. The area difference between RV64 and RV32 is because the TLB is searched by a logical address. The areas of the PTW are 12.1 for RV64 and 6.7 for RV32, regardless of the number of the TLB entries.

Therefore, the PMP occupies a large part of the MMU, which consists of the PTW and TLBs including the PMP.

Table V. AREA OF PMP, TLBS, AND PTW

| | | Area/entry | | | | Area | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | PMP | | TLBs | | PMP | | TLBs | | PTW |
| # of entries | | 8 | 16 | 8 | 64 | 8 | 16 | 8 | 64 | |
| Big | RV64 | 2.8 | 2.8 | | 1.1 | 22.2 | 44.2 | | 68.0 | 12.1 |
| | RV32 | 2.8 | 2.7 | | 1.0 | 22.2 | 43.6 | | 62.8 | 6.7 |
| Mid | RV64 | 2.7 | 2.7 | 2.5 | | 21.7 | 43.7 | 20.2 | | 12.1 |
| | RV32 | 2.7 | 2.7 | 2.2 | | 21.4 | 43.1 | 17.6 | | 6.7 |
| Small | RV64 | 2.0 | 2.0 | | | 15.8 | 32.6 | | | |
| | RV32 | 1.8 | 1.9 | | | 14.4 | 29.7 | | | |

The Rocket tile's TLB is 4-way set associative. Although its area is smaller than a full associative one, it is difficult to support multiple page sizes. So, another array for large page sizes is implemented in addition to the array for 4 KiB pages. This is possible when there are few size variations. Then, the areas per entry for the Big tile are 1.1 for RV64 and 1.0 for RV32.

In the Mid tile, TLBs are 4 entries each with 4 ways, so it is practically full associative, but only the parameters are changed to 4 entries. The areas per entry are 2.5 for RV64 and 2.2 for RV32, which are inefficient. On the other hand, the areas per entry of the PMP are about 2.8 and 2.0 when the VM is enabled and disabled, respectively. The 2.8 is the largest, and even larger than the 2.5 of the inefficient TLB of the Mid tile.

Table VI. F/F RATIO OF EACH MODULE

| | RV64 | | | RV32 | | |
|---|---|---|---|---|---|---|
| | Big | Mid | Small | Big | Mid | Small |
| RocketTile | 33.0 | 42.9 | 40.9 | 40.2 | 43.1 | 40.7 |
| Rocket | 33.5 | 38.9 | 38.4 | 36.2 | 37.5 | 36.4 |
| CSRs | 44.0 | 44.3 | 41.9 | 43.1 | 42.8 | 41.0 |
| Frontend | 46.8 | 52.8 | 54.1 | 47.8 | 52.9 | 54.3 |
| BTB | 46.3 | | | 47.2 | | |
| TLB | 50.7 | 50.7 | | 50.9 | 50.5 | |
| DC | 44.2 | 40.3 | 34.1 | 46.1 | 43.3 | 37.3 |
| TLB | 50.8 | 50.7 | | 50.9 | 49.8 | |
| PTW | 53.0 | 53.3 | | 52.6 | 52.6 | |
| FPU | 20.5 | | | 27.4 | | |
| PMP 8 | 15.8 | 15.9 | 21.9 | 15.5 | 16.1 | 24.0 |
| CSRs | 39.3 | 39.1 | 51.3 | 38.3 | 39.4 | 50.4 |
| PMP 16 | 15.6 | 15.8 | 21.2 | 15.8 | 16.0 | 23.2 |
| CSRs | 39.2 | 39.4 | 49.8 | 39.0 | 39.3 | 50.1 |

Table VI shows the F/F ratio of each module. The smaller the F/F ratio, the larger processing logic per F/F. The F/F ratio of the PMP is very small, which means the processing logic of the PMP is very large. Even the F/F ratio of the FPU, which is a computing intensive unit, is 20 to 27%, whereas the F/F ratio of the PMP is 16% and 21 to 24% when the VM is enabled and disabled, respectively. On the other hand, the TLBs and PTW have the FF ratio exceeding 50%, and the processing logic per FF is small.

## VI. CONCLUSION

We worried about an area overhead problem with a small processor caused by adding the PMP function for the security enhancement because there are various area-increasing factors in comparison with the TLB, and evaluated the influence of adding the PMP on the Rocket tiles.

We confirmed the security enhancement has a large area overhead on a low-end processor core whose MMU occupies a considerable part of its area. As shown in Table IV, the area overhead is a minimum of 4.0% for the Big RV64 tile with PMP=8, and a maximum of 33.3% for the Mid RV32 tile with PMP=16. The area per entry of the PMP reaches to 2.8 when the VM is enabled, whereas that of the TLB is 1.0 to 2.5 depending on its configuration.

The Rocket core is a single-scalar processor with a five-stage pipeline, but there are smaller two- [14] and three-stage [5, 15] pipeline processors for the IoT, and their area overhead of the PMP function must be even more severe.

The PMP is defined a maximum of 16 entries, but some say that it is not enough. However, it is difficult to increase the number of entries efficiently with the current specification checking all entries with arbitrary region sizes reaching to a minimum of 4 bytes on every memory access. Therefore, a specification-level enhancement is necessary to increase the number of entries efficiently.

## ACKNOWLEDGMENT

This paper is based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO) entitled "Research and Development of Fundamental Technology of Secure Open Architecture and its Application to Edge AI." The synthesis library and logic synthesis tool used in this study were provided for chip prototype service at System Design Laboratory of Graduate School of Engineering of the University of Tokyo.

## REFERENCES

[1] A. Waterman, et al., "The RISC-V Instruction Set Manual Volume I: Unprivileged ISA, Version 20190608," RISC-V Foundation, June 8, 2019.

[2] A. Waterman, et al., "The RISC-V Instruction Set Manual, Volume II: Privileged Architecture, Version 1.12-Draft," RISC-V Foundation, Jan. 25, 2020.

[3] https://riscv.org/

[4] https://riscv.org/risc-v-cores/

[5] K. Asanović, et al., "The Rocket Chip Generator," Berkeley Tech. Report No. UCB/EECS-2016-17, April 2016.

[6] https://github.com/chipsalliance/rocket-chip

[7] C. Celio, et al., "The Berkeley Out-of-Order Machine (BOOM): An Industry-Competitive, Synthesizable, Parameterized RISC-V Processor," Berkeley Tech. Report No. UCB/EECS-2015-167, June 2015.

[8] C. Celio, et al., "BOOM v2: an open-source out-of-order RISC-V core," Berkeley Tech. Report No. UCB/EECS-2017-157, Sept. 2017.

[9] J. Bachrach, et al., "Chisel: constructing hardware in a scala embedded language," Proceedings of the 49th Annual Design Automation Conference, pp1216–1225, May 2016.

[10] "Chisel/FIRRTL Hardware Compiler Framework," https://chisel.eecs.berkeley.edu/.

[11] https://chipyard.readthedocs.io/

[12] H. Cook, et al., "Diplomatic Design Patterns: A TileLink Case Study," CARRV'17, Oct. 2017.

[13] https://www.sifive.com/documentation/tilelink/tilelink-spec/

[14] Andes Technology, "AndesCore™ N22," http://www.andestech.com/en/products-solutions/andescore-processors/riscv-n22/

[15] Y. Lee, et al., "Z-scale: Tiny 32-bit RISC-V Systems", OpenRISC Conf., 2015.