International Journal of Networking and Computing – www.ijnc.org, ISSN 2185-2847
Volume 12, Number 1, pages 204–217, January 2022

An Implementation of a Grid Square Codes Generator on a RISC-V Processor

Jubee Tada
Graduate School of Science and Engineering, Yamagata University
Yonezawa, Yamagata, 992-8510, Japan

and

Keiichi Sato
Department of Information Systems, Yamagata College of Industry and Technology
Yamagata, Yamagata, 990-2473, Japan

Received: February 15, 2021
Revised: May 4, 2021
Revised: July 30, 2021
Accepted: August 13, 2021
Communicated by Hideharu Amano

Abstract

This paper implements an execution unit that generates grid square codes from latitude and longitude on a RISC-V processor and evaluates its performance. In recent years, a statistical analysis which uses grid square codes has been focused. Although grid square codes are obtained from latitude and longitude based on several equations, this calculation requires a long computing time because it needs a lot of floating-point instructions.

In this paper, an execution unit which generates grid square codes from latitude and longitude is designed, and the instruction which generates grid square codes by using the unit is implemented on a RISC-V processor. The proposed execution unit can generate the corresponding grid square code from latitude and longitude in one cycle.

As a benchmark, a program that counts the number of times the randomly generated latitude and longitude match the specified grid square code is used. Experimental results show the in-order RISC-V processor with the proposed unit which implemented on an FPGA achieves a 36.6% reduction of execution time compared to the original processor. In addition, the performance evaluation of the out-of-order RISC-V processor with the proposed unit by using the gem5 simulator shows a 23.9% reduction of execution cycles compared to the original processor.

This paper also designs an execution unit which converts grid square codes to latitude and longitude. We call this conversion "Grid-to-Degree conversion". Experimental results show the in-order RISC-V processor with the unit which implemented on an FPGA achieves a 3.65% reduction of execution time compared to the original processor.

Keywords: Floating-point arithmetic, Microprocessors, Field programmable gate arrays, Grid square statistics

1 Introduction

In recent years, a statistical analysis which uses grid square codes has been focused. A statistical analysis using grid square codes is important to clarify the migration history of people and the population density. Although grid square codes are obtained from latitude and longitude based on equations, this calculation requires a long computing time because it needs a lot of floating-point instructions. Therefore, a method which accelerates the generation of grid square codes from latitude and longitude is strongly required.

In this paper, an execution unit which generates grid square codes from latitude and longitude is designed, and the instruction which generates grid square codes by using the unit is implemented on a RISC-V processor [1]. The proposed execution unit can generate the corresponding grid square code from latitude and longitude in one cycle. The RISC-V processor with the proposed unit is implemented on an FPGA, and the performance is evaluated.

This paper also designs an execution unit which converts grid square codes to latitude and longitude in one cycle. As compared to generating grid square codes, the number of instructions which required for converting grid square codes to latitude and longitude is small. Therefore, the computing time which could be reduced by the unit is short. However, because floating-point arithmetic units are not used, the power consumption will be reduced when the unit could be implemented with a small amount of resources.

This paper is organized as follows. Section 2 outlines grid square codes. In Section 3, an execution unit which generates grid square codes from latitude and longitude is proposed. Section 4 evaluates the performance of a RISC-V processor which implements the proposed execution unit. Section 5 concludes this paper.

2 GRID SQUARE CODES

In this study, grid square codes based on JIS X0410 [2] specified as Japanese Industrial Standard (JIS) are used. JIS X0410 defines Japanese procedures for generating grid square statistics for both government of Japan statistics and industrial applications. In Japan, the Statistics Bureau, the Ministry of Internal Affairs and Communications, and the Ministry of Land, Infrastructure, Transport and Tourism provide grid square data for Japanese statistical surveys such as censuses, economic surveys, and censuses, national land numeric information, facilities, natural environment and land usage [3]. Grid square codes are codes given when subdividing landscape into rectangular subregions by latitude and longitude [2]. The codes enable us to identify each grid square as a unique location from latitude and longitude. In addition, the codes are useful for data analysis in grid square statistics. Figure 1 shows a schematic representation of the grid square codes. The codes are expressed by numeric digits in which length corresponds to the spatial resolution, and its notation is defined as "puqvrw". "pu" is called "1st grid code", which is expressed by four numeric digits and 80km grid square code. Also, "puqv" is called "2nd grid code", which is expressed by six numeric digits and divides 1st grid code into 8 parts of equal length and breadth (10km grid square code). Furthermore, "puqvrw" is called "3rd grid code", which is expressed by eight numeric digits and divides 2nd grid code by 10 equally (1km grid square code). Generally, 3rd grid code is called the "Standard Area Grid". Because the JIS X0410 is stipulated for the area around Japan, the range of latitude is 20 to 46 degrees and the range of longitude is 122 to 154 degrees.

For data analysis based on grid square statistics, it is necessary that the data conversion from degree notation of latitude and longitude to the grid square codes. Specifically, the conversion method is denoted as the following equations [2]:

$$p = \lfloor latitude \times 60 \div 40 \rfloor \; (p\, is\, two\, digits) \tag{1}$$

$$a = (latitude \times 60 \div 40 - p) \times 40 \tag{2}$$

$$q = \lfloor a \div 5 \rfloor \; (q\, is\, one\, digit) \tag{3}$$

$$b = (a \div 5 - q) \times 5 \tag{4}$$

$$r = \lfloor b \times 2 \rfloor \; (r\, is\, one\, digit) \tag{5}$$

$$u = \lfloor longitude - 100 \rfloor \; (u\, is\, two\, digits) \tag{6}$$

$$f = longitude - 100 - u \tag{7}$$

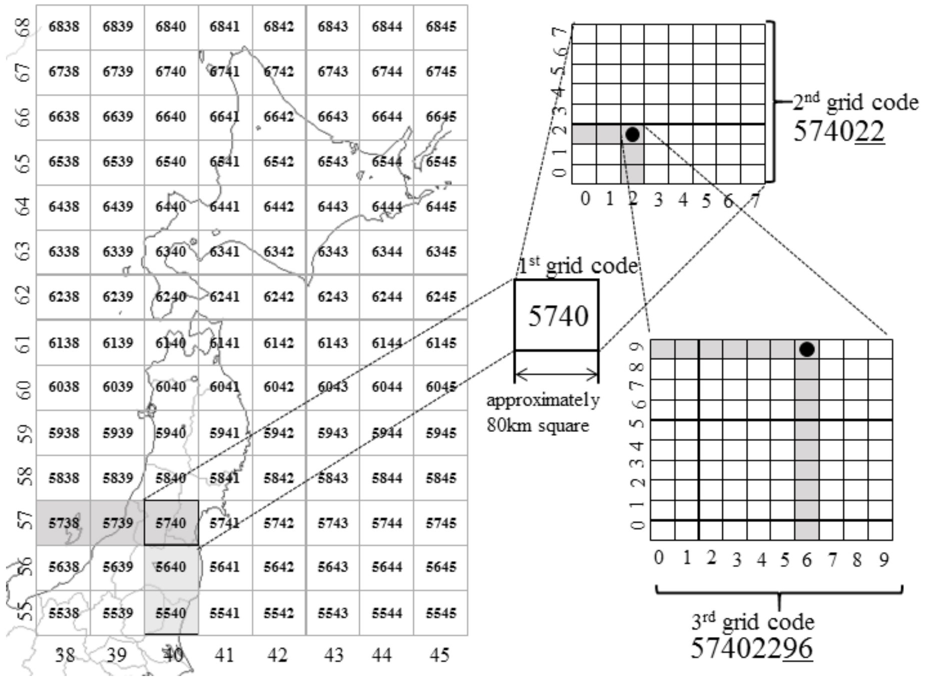


Figure 1: Schematic representation of Grid square codes based on JIS X0410

$$v = \lfloor f \times 8 \rfloor \; (v \; is \; one \; digit) \tag{8}$$

$$g = (f \times 8 - v) \tag{9}$$

$$w = \lfloor g \times 80 \rfloor \; (w \; is \; one \; digit) \tag{10}$$

Here, "$\lfloor \; \rfloor$" denotes the floor function. In this paper, latitude and longitude are expressed in decimal degrees. For example, when latitude and longitude are given as 38.24583 and 140.3313, respectively, 3rd grid code of 57402296 is obtained by using above equations.

Figure 2 shows a schematic representation of an area corresponding to grid square code "57402296". The area includes Yamagata station. In addition, Grid square code "57403206" adjacent to the upper side of the grid square code includes Kajo park.

To output the result obtained by data analysis with the code, visualization on a map is useful. Because we can allow to easily understand distribution such as the population or temperature in each area. To visualize the result, it is necessary data conversion from the grid square codes to the degree notation of latitude and longitude. We call this conversion "Grid-to-Degree conversion". Specifically, the conversion method is denoted as the following equations:

$$W_{latitude} = \frac{2}{3} \tag{11}$$

$$W_{longitude} = 1 \tag{12}$$

$$W'_{latitude} = \frac{W_{latitude}}{8} \tag{13}$$

$$W'_{longitude} = \frac{W_{longitude}}{8} \tag{14}$$

$$W''_{latitude} = \frac{W'_{latitude}}{10} \tag{15}$$

$$W''_{longitude} = \frac{W'_{longitude}}{10} \tag{16}$$

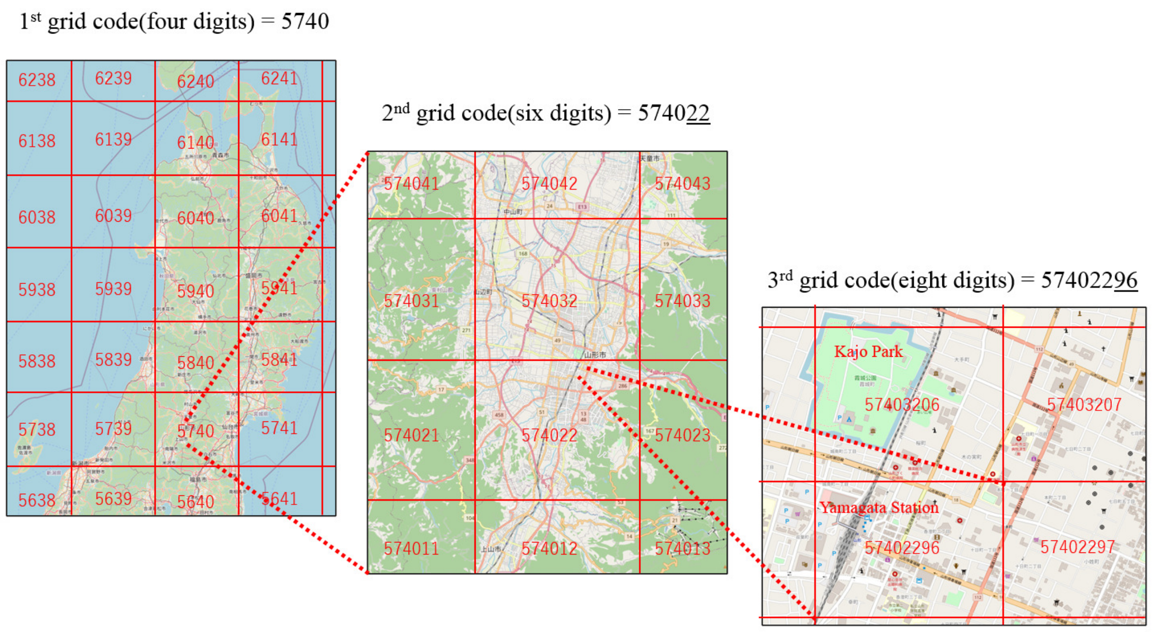


Figure 2: an area corresponding to a grid square code

$$M_{latitude} = \frac{2}{3}p \tag{17}$$

$$M_{longitude} = u + 100 \tag{18}$$

$$M'_{latitude} = M_{latitude} + \frac{\left(\frac{2}{3}q\right)}{8} \tag{19}$$

$$M'_{longitude} = M_{longitude} + \frac{v}{8} \tag{20}$$

$$M''_{latitude} = M'_{latitude} + \frac{\frac{(2/3)r}{8}}{10} \tag{21}$$

$$M''_{longitude} = M'_{longitude} + \frac{\left(\frac{w}{8}\right)}{10} \tag{22}$$

$$M''_{latitude_c} = M''_{latitude} + \frac{W''_{latitude}}{2} \tag{23}$$

$$M''_{longitude_c} = M''_{longitude} + \frac{W''_{longitude}}{2} \tag{24}$$

Here, $W_{latitude}$ and $W_{longitude}$ denote width and height of a 1st grid square code, a concept is shown in Figure 3. $W'_{latitude}$ and $W'_{longitude}$ denote width and height of a 2nd grid square code, $W''_{latitude}$ and $W''_{longitude}$ denote width and height of a 3rd grid square code.

Subsequently, $M_{latitude}$ and $M_{longitude}$ denote latitude and longitude corresponding to southwestern location on a 1st grid square code obtained by Grid-to-Degree conversion (See Figure 3). We call the latitude and the longitude "grid-latitude" and "grid-longitude", respectively. $M'_{latitude}$ and $M'_{longitude}$ denote grid-latitude and grid-longitude corresponding to southwestern location on a 2nd grid square code, $M''_{latitude}$ and $M''_{longitude}$ denote grid-latitude and grid-longitude corresponding to southwestern location on a 3rd grid square code. In addition, $M''_{latitude_c}$ and $M''_{longitude_c}$ denote grid-latitude and grid-longitude corresponding to center location on a 3rd grid square code.

As mentioned above, a lot of operations are required to generate a grid square code from latitude and longitude, and to convert a grid square code to latitude and longitude. Therefore, this paper proposes and evaluates execution units which perform these operations at high speed.

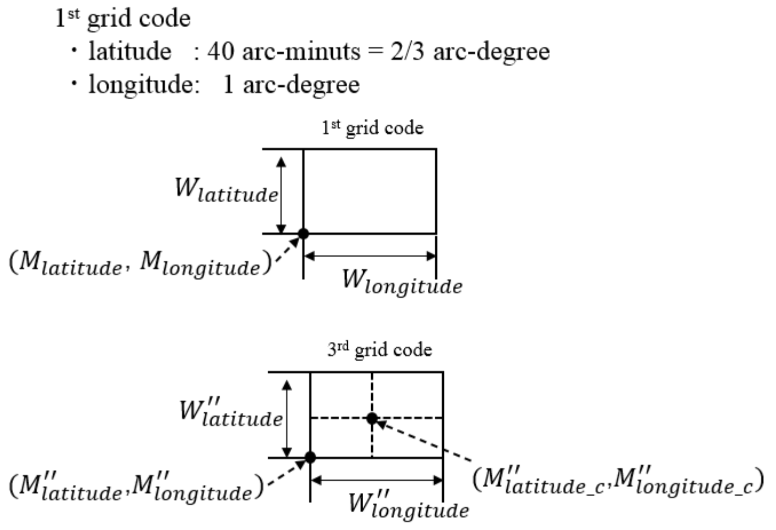

1st grid code
・latitude : 40 arc-minuts = 2/3 arc-degree
・longitude: 1 arc-degree
1st grid code
$W_{latitude}$
$(M_{latitude}, M_{longitude})$
$W_{longitude}$
3rd grid code
$W''_{latitude}$
$(M''_{latitude}, M''_{longitude})$
$(M''_{latitude\_c}, M''_{longitude\_c})$
$W''_{longitude}$

Figure 3: A concept for grid-latitude and grid-longitude

3 Design of a grid square codes generator

Figure 4 shows the function which generates the corresponding grid square code from latitude and longitude in 64-bit RISC-V processors. The function is implemented in C language, compiled by GCC 9.2.0 with an optimization option -O3.

As shown in Figure 4(a), over 50 instructions are required in 64-bit RISC-V processors. In order to generate the corresponding grid square code from latitude and longitude, a lot of instructions are required, and a large part of these instructions is floating-point instructions. Therefore, this paper proposes an execution unit which can generate the corresponding grid square code from latitude and longitude in one cycle.

Figure 5 shows the structure of the proposed grid square code generator. The proposed generator generates the "puqvrw" value described in Section 2 from two single-precision floating-point values which illustrate latitude and longitude.

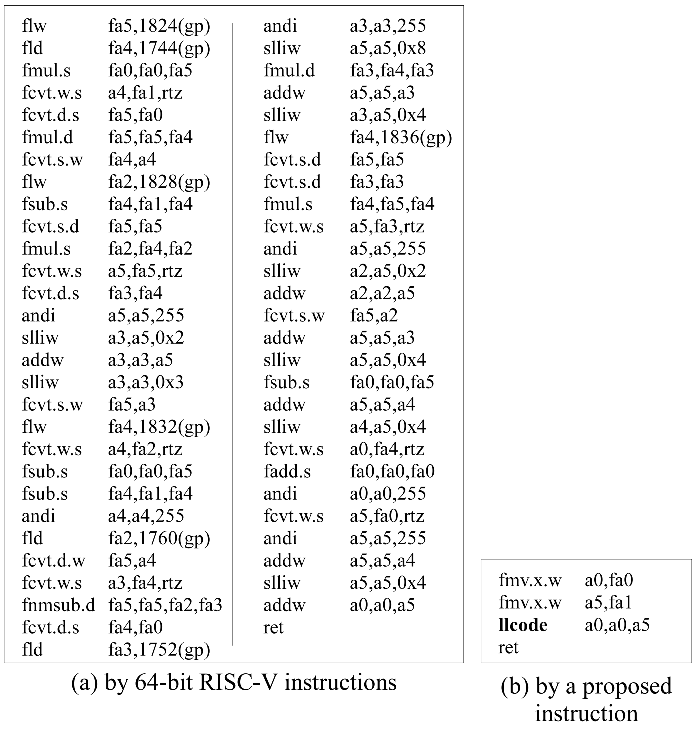
In this paper, each "puqvrw" is expressed as several bits to considering the ease of matching the grid square codes to 1st grid code and 2nd grid code. "p" and "u" are expressed by 8-bit and the others are expressed 4-bit, so the entire "puqvrw" is expressed 32-bit.

The behavior of the proposed generator is as follows. At first, it is confirmed whether the input latitude and longitude are within an appropriate range. The range converted by the grid square codes is 20 to 46 degrees in latitude, and 122 to 154 degrees in longitude. If latitude or longitude is out of range, the output value is set to zero.

Next, "p","q", and "r" are generated by latitude. Figure 6(a) shows the structure of the generator for latitude.

At first, the exponent field of the input latitude is checked. The range of latitude is 20 to 46 degrees, so the exponent value is "10000011" or "10000100". Because it is necessary to align the exponent value, if the value is "10000100", the temporary value is set to the 1-bit left shifted significand of the input latitude. Otherwise, that is set to the significand of the input latitude.

The mechanism to generate "p", "q", and "r" from the temporary value is as follows. The value of "p" is the integer part of the temporary value times 1.5. This value is obtained by adding the temporary value and the 1-bit left shifted temporary value, and taking the upper 7-bits of the result. The value of "q" is upper 3-bit of the remaining of the result. The remaining value after taking "q" is multiplied by 10. This multiplication could be realized by adding the value and the 2-bit shifted value. The value of "r" is obtained by taking upper 4-bit of the result of this addition. As mentioned above, the hardware resources to generate "p", "q" and "r" from the temporary value are one comparator, two adders and one multiplexer.



```
flw       fa5,1824(gp)        andi      a3,a3,255
fld       fa4,1744(gp)        slliw     a5,a5,0x8
fmul.s    fa0,fa0,fa5         fmul.d    fa3,fa4,fa3
fcvt.w.s  a4,fa1,rtz          addw      a5,a5,a3
fcvt.d.s  fa5,fa0             slliw     a3,a5,0x4
fmul.d    fa5,fa5,fa4         flw       fa4,1836(gp)
fcvt.s.w  fa4,a4              fcvt.s.d  fa5,fa5
flw       fa2,1828(gp)        fcvt.s.d  fa3,fa3
fsub.s    fa4,fa1,fa4         fmul.s    fa4,fa5,fa4
fcvt.s.d  fa5,fa5             fcvt.w.s  a5,fa3,rtz
fmul.s    fa2,fa4,fa2         andi      a5,a5,255
fcvt.w.s  a5,fa5,rtz          slliw     a2,a5,0x2
fcvt.d.s  fa3,fa4             addw      a2,a2,a5
andi      a5,a5,255           fcvt.s.w  fa5,a2
slliw     a3,a5,0x2           addw      a5,a5,a3
addw      a3,a3,a5            slliw     a5,a5,0x4
slliw     a3,a3,0x3           fsub.s    fa0,fa0,fa5
fcvt.s.w  fa5,a3              addw      a5,a5,a4
flw       fa4,1832(gp)        slliw     a4,a5,0x4
fcvt.w.s  a4,fa2,rtz          fcvt.w.s  a0,fa4,rtz
fsub.s    fa0,fa0,fa5         fadd.s    fa0,fa0,fa0
fsub.s    fa4,fa1,fa4         andi      a0,a0,255
andi      a4,a4,255           fcvt.w.s  a5,fa0,rtz
fld       fa2,1760(gp)        andi      a5,a5,255
fcvt.d.w  fa5,a4              addw      a5,a5,a4
fcvt.w.s  a3,fa4,rtz          slliw     a5,a5,0x4
fnmsub.d  fa5,fa5,fa2,fa3     addw      a0,a0,a5
fcvt.d.s  fa4,fa0             ret
fld       fa3,1752(gp)
```

(a) by 64-bit RISC-V instructions

```
fmv.x.w   a0,fa0
fmv.x.w   a5,fa1
llcode    a0,a0,a5
ret
```

(b) by a proposed instruction

Figure 4: A function which generates the corresponding grid square code from latitude and longitude

The values of ”p”, ”q”, and ”r” are generated from 38.24583 as follows. In this case, the value of ”p” is 57, the value of ”q” is 2 and the value of ”r” is 9. The exponent value of the 38.24583 is ”10000100”, and the significand value with an implicit leading bit is ”100110001111101110111011”. The temporary value is set to the 1-bit left shifted value, so the value is ”1001100011111011101110110”. The temporary value and the 1-bit left shifted temporary value are added, and the result is ”0111001010111100110011000 10”. The upper 7-bit of the value is ”p”, so it is ”0111001”. The value of ”q” is next 3-bit, so it is ”010”. The remaining of the value is ”11110011001100010”. The value and the 2-bit left shifted value is added, and the result is ”1001011111111111010100”. The value of ”r” is the upper 4-bit of the result, so it is ”1001”.

Next, ”u”, ”v”, and ”w” are generated by longitude. Figure 7 shows the structure of the generator for longitude. At first, the exponent field of the input longitude is checked. The range of longitude is 122 to 154 degrees, so the exponent value is ”10000101” or ”10000110”. If the value is ”10000110”, the temporary value is set to the 1-bit left shifted significand of the input longitude. Otherwise, that is set to the significand of the input longitude.

The mechanism to generate ”u”, ”v”, and ”w” from the temporary value is as follows. The value of ”u” is obtained by subtracting 100 from the upper 8-bit of the temporary value. The value of ”v” is upper 3-bit of the remaining of the temporary value. The remaining value after taking ”v” is multiplied by 10. This multiplication could be realized by adding the value and the 2-bit shifted value. The value of ”w” is obtained by taking upper 4-bit of the result of this addition. As mentioned above, the hardware resources to generate ”u”, ”v” and ”w” from the temporary value are one comparator, one subtractor, one adder and one multiplexer.

The values ”u”, ”v”, and ”w” are generated from 140.3313 as follows. In this case, ”u” is 40, ”v” is 2 and ”w” is 6. The exponent value of the 140.3313 is ”10000110”, and the significand value with an implicit leading bit is ”100011000101010011010000”. The temporary value is set to the 1-bit left shifted value, so the value is ”1000110001010100110100000”. The upper 8-bit of temporary value is ”10001100”, and 100 is subtracted from it. The result is ”00101000”, and it is the value of ”u”. The remaining of the temporary value is ”01010100110100000”. the value of ”v” is the upper 3-bit

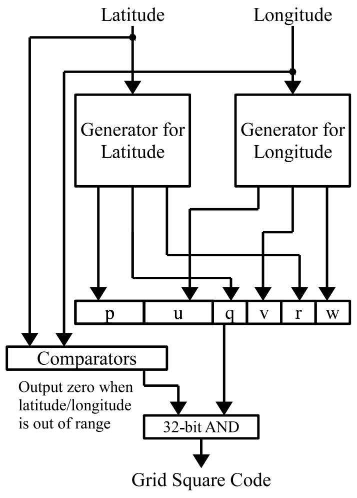


Figure 5: Structure of the proposed grid square codes generator

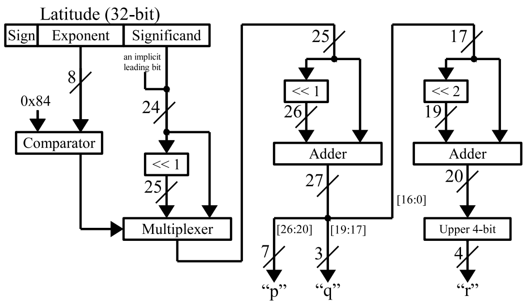


Figure 6: Structure of the generator for latitude

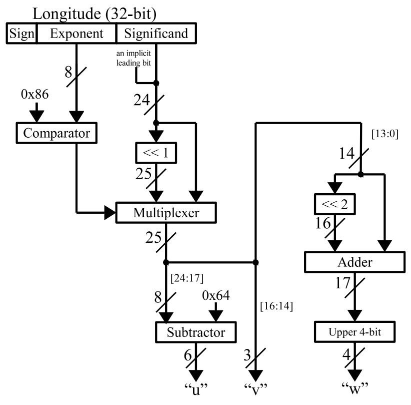


Figure 7: Structure of the generator for longitude

of the value, so it is "010". The remaining value after taking "v" is "10100110100000". The value and the 2-bit left shifted value is added, and the result is "01101000000100000". The value of "w" is the upper 4-bit of the result, so it is "0101".

In this paper, the proposed generator is implemented on the ALU of a RISC-V processor. As shown in Figure 4(b), the RISC-V processor which implements the proposed generator can generate the corresponding grid square code from latitude and longitude by three instructions. In Figure 4(b), "llcode" is the instruction which generates the corresponding grid square code from latitude and longitude. Although the generation can be done by one instruction, additional two instructions are need to move the two values from f registers to x registers.

4 Design of a Grid-to-Degree converter

Figures 8 and 9 show the functions which convert the grid square code to latitude and longitude in 64-bit RISC-V processors, respectively. As shown in Figure 8(a) and Figure 9(a), about 20 instructions are required in 64-bit RISC-V processors for each conversion. As compared to the grid square codes generation, Grid-to-Degree conversion requires small number of instructions, and a large part of these instructions are integer instructions. Although the performance of a Grid-to-Degree converter is lower than that of the grid square codes generator, it is considered that the converter will reduce the computing time. In addition, the converter is useful when floating-point units are not implemented on a processor like as the edge computing. Therefore, this paper proposes a Grid-to-Degree converter which can convert the grid square code to latitude and longitude.

Figure 10 shows the structure of the proposed Grid-to-Degree converter. The proposed converter converts the "puqvrw" value described in Section 2 into two single-precision floating-point values which illustrate latitude and longitude. The behavior of the proposed converter is as follows. At first, it is confirmed whether the input "p" and "u" values are within an appropriate range. The range of "p" is 30 to 68, and the range of "u" is 22 to 53. If "p" or "u" is out of range, the output values are set to zero.

Next, "p","q", and "r" are converted to latitude. Figure 11 shows the structure of the converter for latitude. The $M_{latitude}$ is calculated by using the following equation from equations (17), (19) and (21):

$$M_{latitude} = ((p \times 80) + (q \times 10) + r) \div 120 \tag{25}$$

At first, to multiply "p" by 80, "p" and the 2-bit left shifted "p" are added, and the result is 4-bit left shifted. Also to multiply "q" by 10, "q" and the 2-bit left shifted "p" are added, and the result is 1-bit left shifted. The multiplied "p", the multiplied "q", and "r" are added, and the result is multiplied by a significand of 1/120. Next, "p" is checked for deciding the exponent value and the significand value. When "p" is less than 48, the exponent value is set to "10000011", and the significand value is obtained by taking 33th to 11th bits of the result of multiplication. When "p" is 48 or more, the exponent value is set to "10000100", and the significand value is obtained by taking 34th to 12th bits of the result of multiplication. As mentioned above, the hardware resources to convert "p", "q" and "r" to latitude are three adders, one multiplier, one comparator and two multiplexers.

Next, "u", "v", and "w" are converted to longitude. Figure 12 shows the structure of the converter for longitude. The $M_{longitude}$ is calculated by using the following equation from equations (18), (20) and (22):

$$M_{longitude} = ((u \times 80) + (v \times 10) + w) \div 80 \tag{26}$$

At first, 100 is added to "u". To multiply "u+100" by 80, the result and the 2-bit left shifted "u+100" are added, and the result is 4-bit left shifted. Also to multiply "v" by 10, "q" and the 2-bit left shifted "v" are added, and the result is 1-bit left shifted. The multiplied "u", the multiplied "v", and "w" are added, and the result is multiplied by a significand of 1/80. Next, "u" is checked for deciding the exponent value and the significand value. When "u" is less than 28, the exponent value is set to "10000101", and the significand value is obtained by taking 35th to 13th bits of the result of multiplication. When "u" is 28 or more, the exponent value is set to "10000110", and the significand value is obtained by taking 36th to 14th bits of the result of multiplication. As mentioned above, the hardware resources to convert "p", "q" and "r" to latitude are four adders, one multiplier, one comparator and two multiplexers.

The proposed Grid-to-Degree converter is implemented on the ALU of a RISC-V processor. As shown in Figure 8(b) and Figure 9(b), the RISC-V processor which implements the proposed generator can convert the grid square code to latitude and longitude by two instructions. In Figure 8(b), "codelat" is the instruction which converts the grid square code to latitude. In Figure 9(b), "codelong" is the instruction which converts the grid square code to longitude. Although these conversions can be done by one instruction, additional one instruction is need to move one value from x register to f register. Therefore, the grid square code is converted to latitude and longitude by four instructions.

5 Performance Evaluation

To evaluate the performance of the proposed units, these are implemented on a RISC-V processor. As RISC-V is open architecture which enables open-source hardware implementations, there are a lot of instruction extension examples [4]. So a RISC-V processor is selected as a baseline processor in this paper.

In the implementation on FPGA, Ariane [5] is used as a baseline processor. Ariane is a 64-bit, single-issue, in-order RISC-V processor. In the experiments, the digilent genesys2 board which equips Xilinx XC7K325T-2FFG900C FPGA is used. The clock frequency when Ariane is implemented on this board is 50MHz. Xilinx Vivado Design Suite 2019.2 is used for designing.

As a result of logic synthesis, the proposed generator uses 104 LUTs, and the maximum path delay is 6.956ns. The proposed converter uses 69 LUTs and 2 DSPs, and the maximum path delay is 10.223 ns.

Table 1 shows the hardware utilization of the original Ariane processor, the processor with the proposed generator, and the processor with the proposed converter. As compared to the baseline processor, the processor with the proposed generator increases the number of LUTs by 172 and the number of FFs by 4, and the processor with the proposed converter increases the number of LUTs by 421, and the number of FFs by 5.

The maximum path delay of the processor with the proposed generator and the processor with the proposed converter are 16.753 ns and 16.026 ns, respectively, while it is 16.307 ns for the baseline processor. These are short enough to achieve a 50MHz clock frequency. In all cases, the maximum path delay is observed on the scoreboard.

To evaluate the performance of the proposed generator, the benchmark is executed on linux, and the execution time is measured. As the benchmark, a program that counts the number of times the randomly generated latitude and longitude matches the specified grid square code is used. The number of iterations is 10 million, and the number of grid square codes for matching is 16. Figure 13 shows the experimental results. As compared to the baseline processor, the processor with the proposed generator achieves a 36.6% reduction of the execution time.

Ariane is an in-order, single issue processor. Therefore, it is considered that the effect of the proposed generator is enhanced. In order to evaluate the proposed generator in an out-of-order processor, the gem5 simulator [6] is used. Table 2 shows the parameter of the simulated processor. The clock frequency is set to 1.7GHz, that is the clock frequency when Ariane processor is designed with 22 nm FD-SOI process [5].

Figure 14 shows the simulation results. As compared to the baseline, the proposal achieves a 23.9% reduction of the number of execution cycles. Figure 15 shows the breakdown of instructions executed in the baseline and the proposal. As compared to the baseline, the proposal achieves a 39.5% reduction of the number of instructions. This is because the number of instructions required to generate the grid square code has been reduced. Compared to the reduction of the number of instructions, the reduction of the number of execution cycles is small. It is considered to be

```
srliw     a2,a0,0x18
srliw     a4,a0,0xc
andi      a3,a4,15
slliw     a5,a2,0x2
addw      a5,a5,a2
slliw     a4,a3,0x2
srliw     a0,a0,0x4
andi      a0,a0,15
addw      a4,a4,a3
slliw     a5,a5,0x4
addw      a5,a5,a0
slliw     a4,a4,0x1
addw      a5,a5,a4
fcvt.d.w  fa0,a5
fld       fa5,1808(gp)
fmul.d    fa0,fa0,fa5
fcvt.s.d  fa0,fa0
ret
```

(a) by 64-bit RISC-V instructions

```
codelat   a1, a0
fmv.w.x   fa0, a1
ret
```

(b) by a proposed instruction

Figure 8: A function which converts the grid square code to latitude

```
srliw     a5,a0,0x10
andi      a2,a5,255
srliw     a4,a0,0x8
slliw     a5,a2,0x2
andi      a3,a4,15
addw      a5,a5,a2
lui       a2,0x2
slliw     a4,a3,0x2
addiw     a2,a2,-192
andi      a0,a0,15
addw      a0,a0,a2
addw      a4,a4,a3
slliw     a5,a5,0x4
addw      a5,a5,a0
slliw     a4,a4,0x1
addw      a5,a5,a4
fcvt.d.w  fa0,a5
fld       fa5,1800(gp)
fmul.d    fa0,fa0,fa5
fcvt.s.d  fa0,fa0
ret
```

(a) by 64-bit RISC-V instructions

```
codelong  a1, a0
fmv.w.x   fa0, a1
ret
```

(b) by a proposed instruction

Figure 9: A function which converts the grid square code to longitude

Table 1: Hardware Utilization

	Baseline	with Generator	with Converter
Total LUTs	64013	64185	64434
Logic LUTs	62421	62593	62842
LUTRAMs	1224	1224	1224
SRLs	368	368	368
FFs	48707	48711	48712
RAMB36	49	49	49
RAMB18	2	2	2
DSP48 Blocks	27	27	29

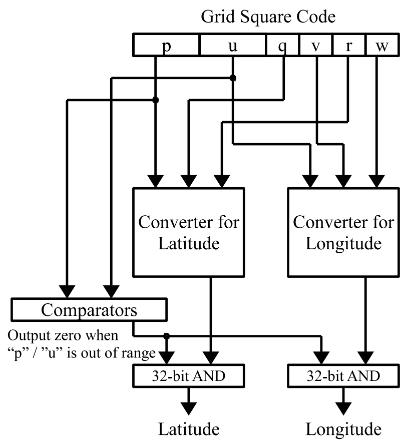


Figure 10: Structure of the proposed grid square codes converter

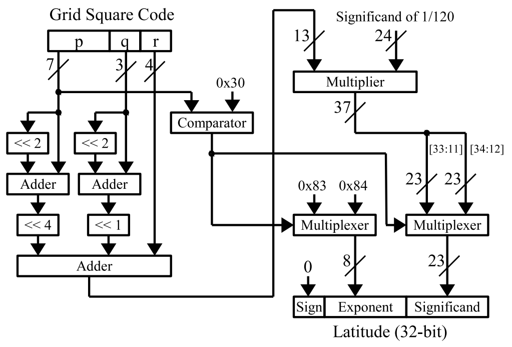


Figure 11: Structure of the converter for latitude

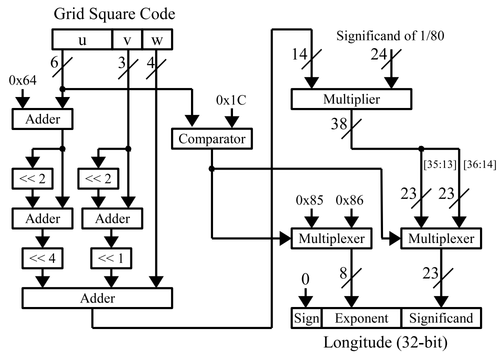


Figure 12: Structure of the converter for longitude

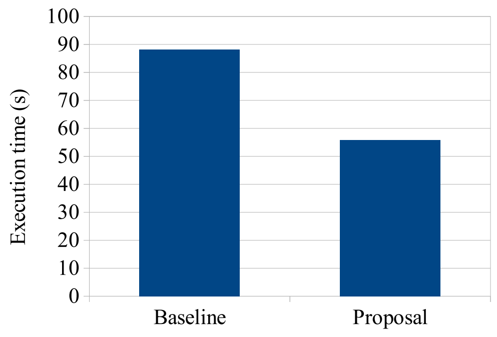


Figure 13: Execution time of the benchmark in processors on FPGA (grid square code generator)

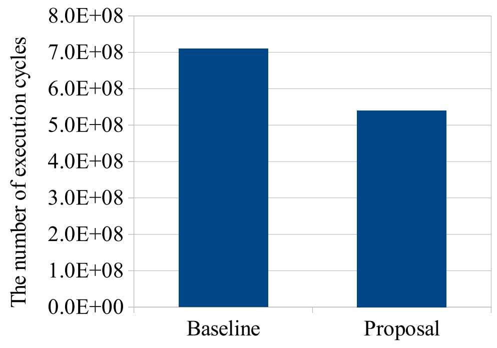


Figure 14: The number of execution cycles of the benchmark in the gem5 simulator (grid square code generator)

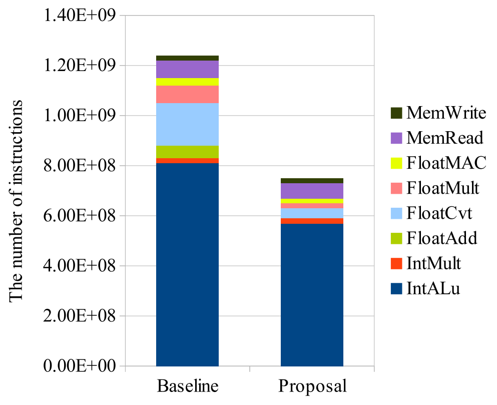


Figure 15: The breakdown of instructions

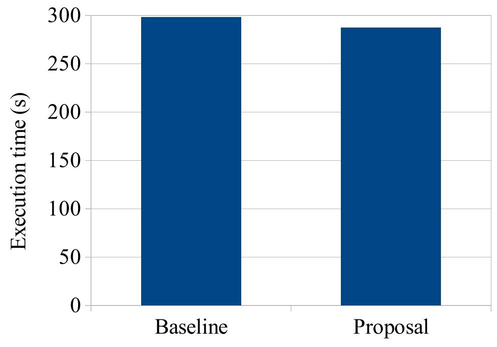


Figure 16: Execution time of the benchmark in processors on FPGA (Grid-to-Degree converter)

Table 2: Parameters of the simulated processor

	Parameter
Clock	1.7GHz
L1I/D size	32KB
L1I/D assoc.	8
L1I/D latency	2
L2 size	256KB
L2 assoc.	8
L2 latency	20

the effect of out-of-order execution. Although the performance improvement rate becomes smaller than the FPGA implementation, the effect of the proposed generator is still large. To evaluate the performance of the proposed converter, the benchmark is executed on linux, and the execution time is measured. As the benchmark, a program that counts the number of times the randomly generated grid square code is near on the specified latitude and longitude is used. The number of iterations is 10 million, and the number of latitudes and longitudes for matching is 16. The benchmark counts the number when the differences between the specified latitude and longitude and the converted latitude and longitude are within 0.01. Figure 16 shows the experimental results. As compared to the baseline processor, the processor with the proposed converter achieves a 3.65% reduction of the execution time. The performance improvement is smaller than that of the proposed generator. This is because the number of instructions required for converting grid square codes to latitude and longitude is smaller than that for generating grid square codes from latitude and longitude.

6 Conclusions

This paper implements an execution unit that generates grid square codes from latitude and longitude on a RISC-V processor and evaluates its performance.

Experimental results show the in-order RISC-V processor with the proposed unit which implemented on an FPGA achieves a 36.6% reduction of execution time compared to the original processor. In addition, the performance evaluation of the out-of-order RISC-V processor with the proposed unit by using the gem5 simulator shows a 23.9% reduction of execution cycles compared to the original processor.

This paper also designs an execution unit which converts grid square codes to latitude and longitude in one cycle. Experimental results show the in-order RISC-V processor with this unit which implemented on an FPGA achieves a 3.65% reduction of execution time compared to the original processor.

Since grid square codes based on JIS X0401 covers only the area around Japan, the available area is limited. To solve this problem, the world grid square code has been proposed, which is an extension of the JIS X0401 and covers the entire world [3]. In future work, an execution unit which generates world grid square codes from latitude and longitude will be designed and evaluated. In addition, an accelerator which speeds up of the matching of grid square codes will be designed and evaluated.

Acknowledgment

This work was partially supported by JSPS KAKENHI Grant Number JP19H02134.

References

[1] A. Waterman and K. Asanovic, ”The RISC-V Instruction Set Manual Volume I: User-Level ISA Document Version2.2,” RISC-V Foundation, 2017.

[2] ”Statistics Bureau of Japan,” Method of Demarcation for Grid Square, [online] Available: http://www.stat.go.jp/english/data/mesh/05.html.

[3] A. Sato, S. Nishimura and H. Tsubaki, ”World Grid Square codes: Definition and an example of World Grid Square data,” 2017 IEEE Int. Conf. on Big Data (BIGDATA), pp. 4156-4165, Dec. 2017.

[4] M. Gautschi et al., ”Near-Threshold RISC-V Core With DSP Extensions for Scalable IoT Endpoint Devices,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 25(10), pp. 2700–2713, 2017.

[5] F. Zaruba and L. Benini, ”The Cost of Application-Class Processing: Energy and Performance Analysis of a Linux-Ready 1.7-GHz 64-Bit RISC-V Core in 22-nm FDSOI Technology,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 29(11), pp. 2629–2640, 2019.

[6] N. Binkert et al. ,”The gem5 simulator,” ACM SIGARCH Computer Architecture News, vol. 39(2), pp. 1–7, 2011.