ICIC Express Letters
Volume **17**, Number **9**, September **2023**
 ISSN 1881-803X
pp. **1029–1037**

HIGH-SPEED PROCESSING OF SHORTEST PATH MAZE SEARCH BY MITIGATION METHOD HARDWARE

Shunsuke Inoue and Naohiko Shimizu*

School of Information and Telecommunication Engineering
Tokai University
2-3-23, Takanawa, Minato-ku, Tokyo 108-8619, Japan
is382405@hope.tokai-u.jp; *Corresponding author: nshimizu@keyaki.cc.u-tokai.ac.jp

Received January 2023; accepted March 2023

Abstract. *We have designed hardware that improves the accuracy of the evaluation of the positional energy of the shortest path maze search. The results of tests on mazes with various patterns showed that the method was 37.2 to 57.8 times faster for 16 ∗ 8 mazes and 46.6 to 75.8 times faster for 16 ∗ 16 mazes, and 56 to 156 times faster for 32 ∗ 16 mazes. The accuracy of the evaluation of positional energy improved.*
Keywords: FPGA, Hardware, Gauss-Seidel method, Parallel processing, High-speed processing, Shortest path search, Maze seach

1. **Introduction.** The path planning method is to find the shortest path between two locations. We use the shortest route search for logistics. The car navigation system uses route finding. VLSI (Very-large-scale integration) studies the use of pathfinding [1]. The world uses path planning in many different areas. Reinforcement learning is a shortest-path search method. Often unsupervised learning in mazes fails the search [7]. Unsupervised learning works based on experience. Unsupervised learning in maze search replicates good experiences. Exploring the maze is not a good experience. Maze search for the shortest path requires planning. The Dijkstra method is a path-planning method [2]. The Dijkstra method is a sequential process. The Dijkstra method is fast in path planning. The Dijkstra method only works in sequential processing. There is a method based on breadth-first search [8]. Breadth-first search is a parallel process [8]. Breadth-first search is not comparable to the Dijkstra method. The GEMAR method is a parallel process [4]. The GEMAR method is not comparable to the Dijkstra method. The mitigation method searches for the shortest path. The mitigation method is maze shortest path planning. Mitigation methods are faster due to parallel processing. We have parallelized the mitigation method on FPGA. Murata and Mitani compared the A* algorithm with the Dijkstra method [5]. Yuriko et al. have accelerated pathfinding using the HDA* method [9]. The HDA* method uses parallel processing for high-speed processing [9]. A* method moves faster than Dijkstra method [5, 9]. There are papers comparing the Dijkstra method [6]. There was a comparative study of search methods [6]. The Dijkstra method is slow with a simple maze [6]. We measured the processing of the Dijkstra method. We measured the processing of the mitigation method. The processing times of the two were compared [3]. Accuracy was further improved in this study. Our contribution was to compare the Dijkstra and mitigation methods. We propose a mitigation method for speeding up. As a result, the proposed method was effective. In Chapter 2, we wrote about the maze search problem. In Chapter 3, we wrote about the theory of mitigation. In Chapter 4, we wrote about parallel processing of mitigation methods. We discussed the verification results in Chapter 5. We wrote the summary in Chapter 6.

DOI: 10.24507/icicel.17.09.1029

2. **Shortest Path Maze Search Problem.** The input to the maze search problem in Figure 3 is map information. Next, the system calculated and output the routes for the start and end points of the map. The maze is a rectangular map of horizontal W squares ∗ vertical H squares. The maze consists of four types of information: walls and roads, starts, and goals. As shown in Figure 1, the point marked by the circle is the starting point. The point marked by the square is the goal point. The walls are gray and the roads are white. We considered finding the shortest path from the start point to the end point on the map in Figure 1. Specify an address from the top left to bottom right for each square in the map in Figure 1. The mitigation method calculates the potential energy value of each cell as shown in Figure 2. Compute the potential energy from the goal for each square as shown in Figure 2. The steepest gradient method computes the path. In the maze, the current location can only go in 4 directions. The steepest descent method is the method of moving in the direction of lower potential energy in Figure 5. The initial value of potential energy is the road with the maximum start value, target 0, and median as shown in Figure 4. Calculate the Euclidean distance using the Gauss-Seidel method. The outermost square of the maze is the wall.

	0	1	2	· · ·	W-3	W-2	W-1
0	0	1	2	· · ·	W-3	W-2	W-1
1	W+0	W+1	W+2	· · ·	W+(W-3)	W+(W-2)	W+(W-1)
2	2W+0	2W+1	2W+2	· · ·	2W+(W-3)	2W+(W-2)	2W+(W-1)
	·	·	·	· · ·	·	·	·
	·	·	·	· · ·	·	·	·
	·	·	·	· · ·	·	·	·
H-3	(H-3)W+0	(H-3)W+1	(H-3)W+2	· · ·	(H-3)W+(W-3)	(H-3)W+(W-2)	(H-3)W+(W-1)
H-2	(H-2)W+0	(H-2)W+1	(H-2)W+2	· · ·	(H-2)W+(W-3)	(H-2)W+(W-2)	(H-2)W+(W-1)
H-1	(H-1)W+0	(H-1)W+1	(H-1)W+2	· · ·	(H-1)W+(W-3)	(H-1)W+(W-2)	(H-1)W+(W-1)

FIGURE 1. Map size

-1(∞)	-1(∞)	-1(∞)	-1(∞)	-1(∞)
-1(∞)	10	13	15	-1(∞)
-1(∞)	8	-1(∞)	-1(∞)	-1(∞)
-1(∞)	6	3	0	-1(∞)
-1(∞)	-1(∞)	-1(∞)	-1(∞)	-1(∞)

FIGURE 2. Map potential energy

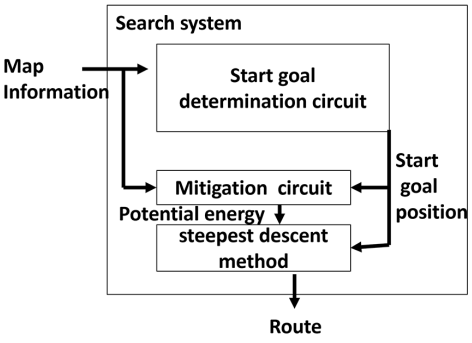

FIGURE 3. Block diagram of maze search system

	Start	Goal	Road	Wall
Init Value	MAX	0	Middle	-1(∞)
Mark				

FIGURE 4. Initial potential energy

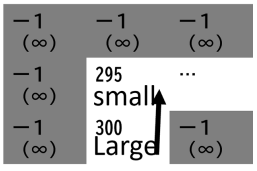

FIGURE 5. Route selection

3. **Mitigation Method.** Mitigation methods include opening the road. When performing the road opening method, the system also calculates the potential energy. In Table 1, the input gear from the starting goal opens up the squares of the path (Figure 6). The road release method ends when the squares release from both the start and finish lines. We then wall the path surrounded by paths in only one direction. We then perform the calculation until the position-energy difference is zero. The mitigation method uses the Gauss-Seidel method to estimate potential energy. In Figure 8, n with the black circle is the potential energy at point n, and Y_n is the path mean potential energy around point n. Also, X_n does not calculate fixed value walls, start points, and goal points. Also, X_n does not calculate or update at fixed walls, starting points, and goal points. Y_n is the average of the maximum and minimum values of the non-wall values (start and goal points). Also, the mitigation method does not calculate the potential energy of the starting point, goal

Table 1. Road opening method

Road opening method
1. The walls except for the start goal of the map represent the walls.
2. Find the street number that is the way on the map entered.
3. We make a street from a wall if the number we find surrounded by a street is not a street and is in one direction.
4. Check to see if the paths connect from both the start and finish lines.
5. If it connects, go to the next step. If it does not connect, go back to 2.
6. If there is no wall around the road in one direction, enclose it with a wall.

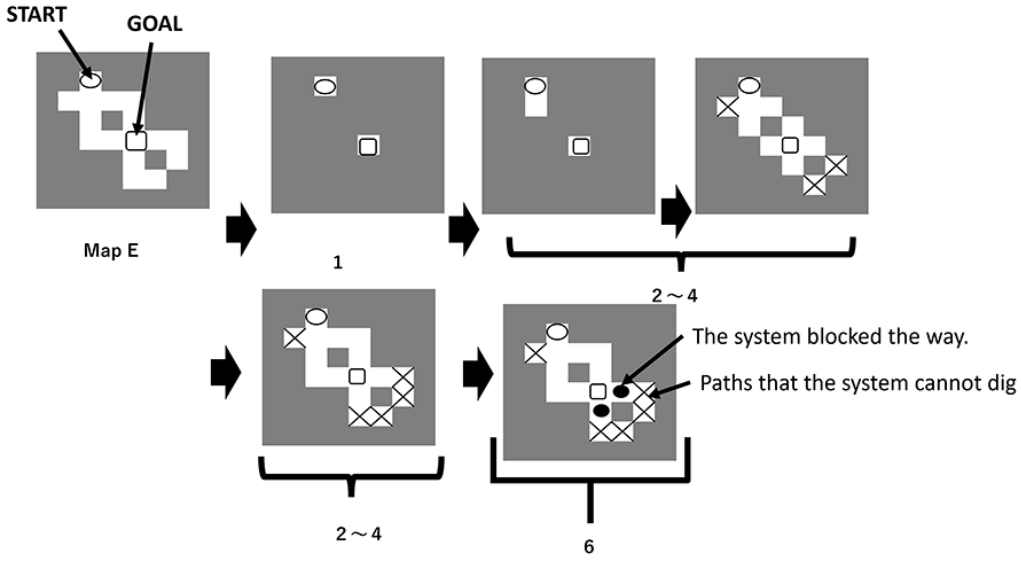

Figure 6. Opening of a road

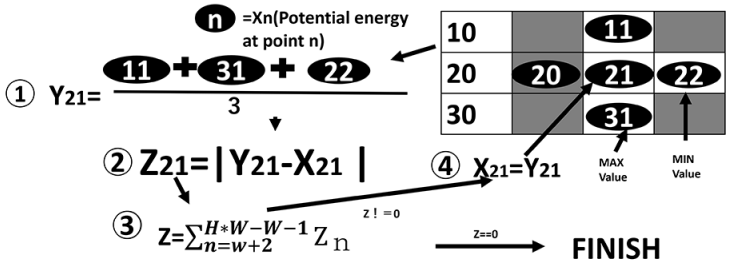

Figure 7. Calculated potential energy (before)

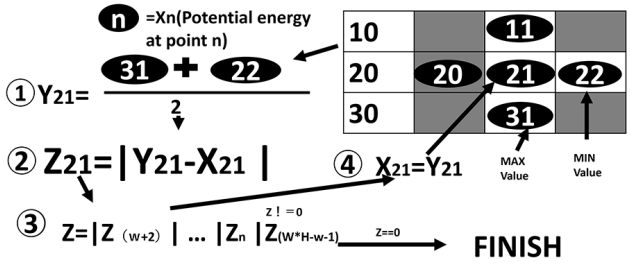

FIGURE 8. Calculated potential energy (after)

WALL「0」	WALL「1」	WALL「0」	WALL「1」	WALL「0」
WALL「1」	LOAD「0」	LOAD「1」	START「0」	WALL「1」
WALL「0」	LOAD「1」	WALL「0」	WALL「1」	WALL「0」
WALL「1」	LOAD「0」	LOAD「1」	GOAL「0」	WALL「1」
WALL「0」	WALL「1」	WALL「0」	WALL「1」	WALL「0」

FIGURE 9. Gauss-Seidel method

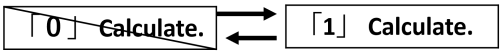

FIGURE 10. Gauss-Seidel method caculation plot

point, and wall. The mitigation method finds the average X of cells that are not adjacent to each cell. By finding the average X, you can search for the shortest path maze with the steepest descent method. In the steepest descent method, after calculating the potential energy of each square as shown in Figure 5, it proceeds in the direction with the lowest potential energy. It also reduces the number of calculations and the order of calculations. The Gauss-Seidel method obtains the average Y_n of the potential energies of "0" and "1" as shown in Figure 9 and Figure 10. The mitigation method calculates potential energy in the following process. The Gauss-Seidel method uses a single core to calculate two adjacent squares alternately, as shown in Figures 9 and 10. Previously, in the case of a non-wall in four directions and three or more directions, as in Figure 7, the average value of the non-wall squares was Y_n. However, to improve accuracy, the position energy was evaluated as the average of the maximum and minimum values in the four directions, as shown in Figure 8.

1) For the potential energy X_{11} at address 11 in Figure 8, find the average value Y_{11} of the non-wall positions X_{12} and X_{21}. It also finds the average Y_n of potential energies around the potential energies X_n of other paths. Two or more non-wall values near 11 (near n)
$$Y_{11} = \frac{X_{21} + X_{12}}{2} \tag{1}$$
2) Find the absolute value Z_{11} of the potential energy difference between X_{11} and Y_{11}. It also finds the difference Z_n between the absolute values of the potential energies of X_n and Y_n (path position n).
$$Z_{11} = |Y_{11} - X_{11}| \tag{2}$$
3) Find the sum Z of all Z_n values.
$$Z = \sum_{n=0}^{(W*H-1)/2} Z_n \tag{3}$$

4) If Z is non-zero, rewrite X_n as Y_n and do the same from step 1).

$$X_{11} = Y_{11} \tag{4}$$

In Formula (1) of step 1), find the average Y_n of the non-wall values (starting point, goal point, road) around point n. In Formula (2) of step 2), find the absolute value Z_n of the X_n difference value of Y_n. In Formula (3) of step 3), find the total Z of Z_n in all cells. In Formula (1) of step 1), find the average Y_n of the maximum and minimum non-wall surface values (start point, goal point, road) around point n. In Formula (2) of step 2), find the average Y_n of the maximum and minimum non-wall surface values (start point, goal point, road) around point n. In Formula (4) of step 4), if Z is 0 or more, replace X_n with Y_n. If Z is 0, the mitigation method ends and the steepest descent method performs the route maze search. Iterating calculates potential energy X_n until Z reaches 0.

4. **Parallel Processing of Mitigation Methods.** The black circle n in Figure 14 is the potential energy at address n. The mitigation hardware performs parallel processing in the circuit and produces an average Y_n of non-wall potential energy around the address n. The mitigation method performs parallel processing, as shown in Figures 15, 11, 12 and 13. Then run the pipeline process for the two clocks split in Figure 16 as shown in

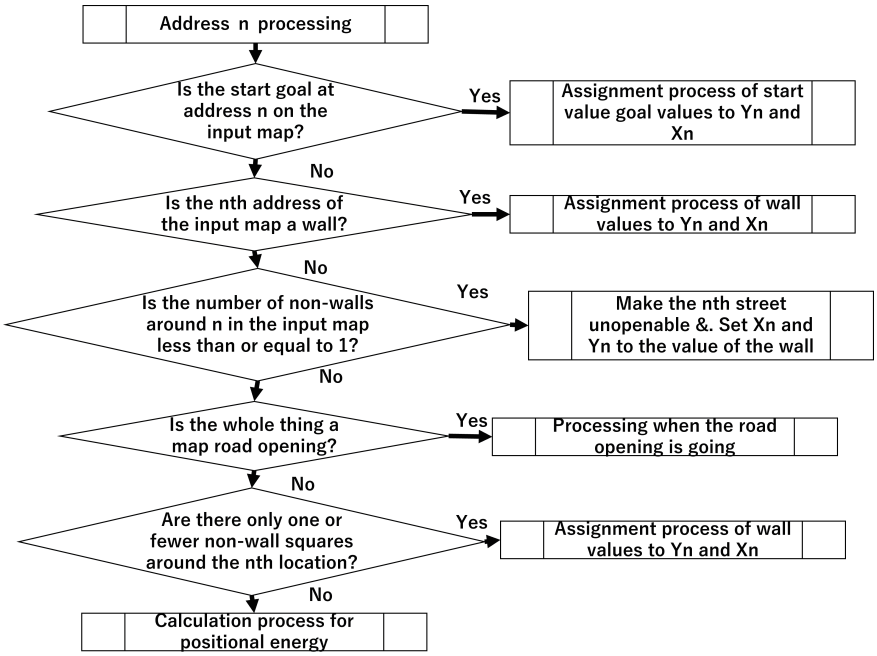

FIGURE 11. Processing of each square

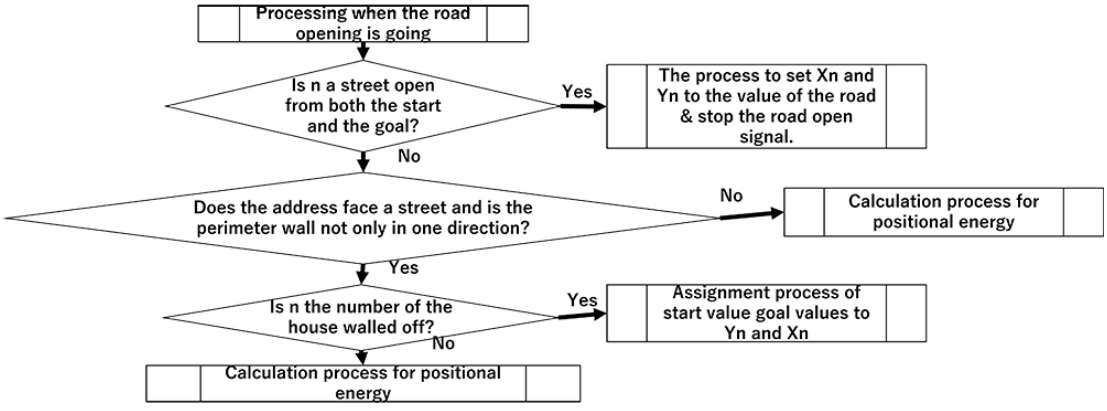

FIGURE 12. Processing when the road opening

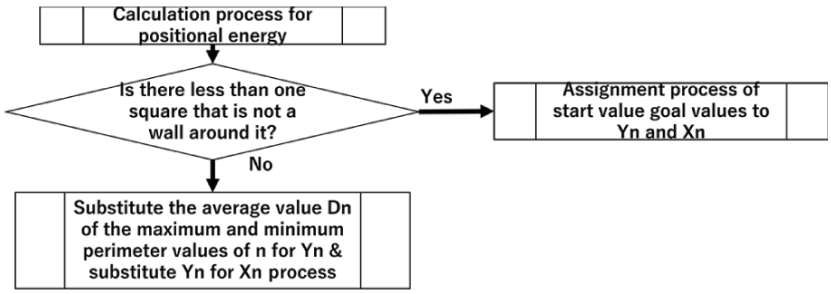

FIGURE 13. Calculation process for positional energy

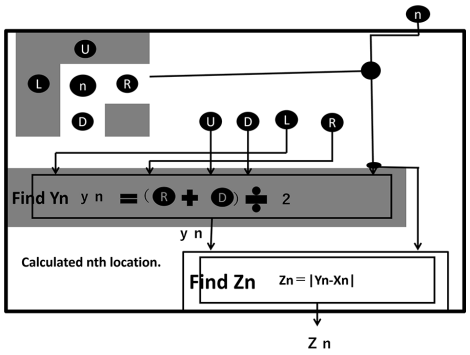

FIGURE 14. Calculation address n

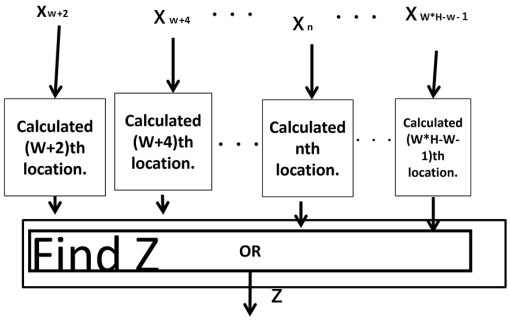

FIGURE 15. Parallel processing

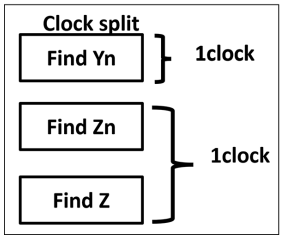

FIGURE 16. Clock split

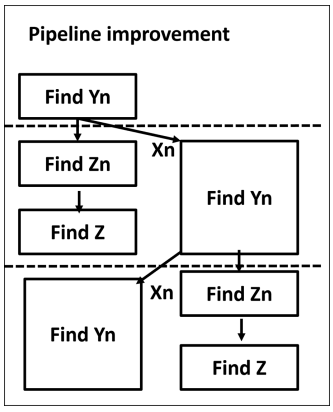

FIGURE 17. Pipeline processing

Figure 17 to calculate the previous Z while calculating the average Y_n of the values not displayed on the wall.

5. **Verification and Comparison.** When the map is large, parallel operations increase, and the FPGA mitigation method is faster than the Central Processing Unit (CPU) Dijkstra method. We verified the algorithms of the $16 * 8$ square map of Figure 20, the

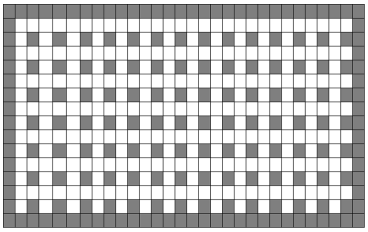

FIGURE 18. Map $32 * 16$

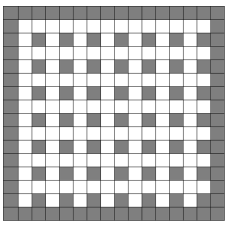

FIGURE 19. Map $16 * 16$

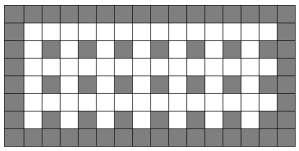

FIGURE 20. Map $16 * 8$

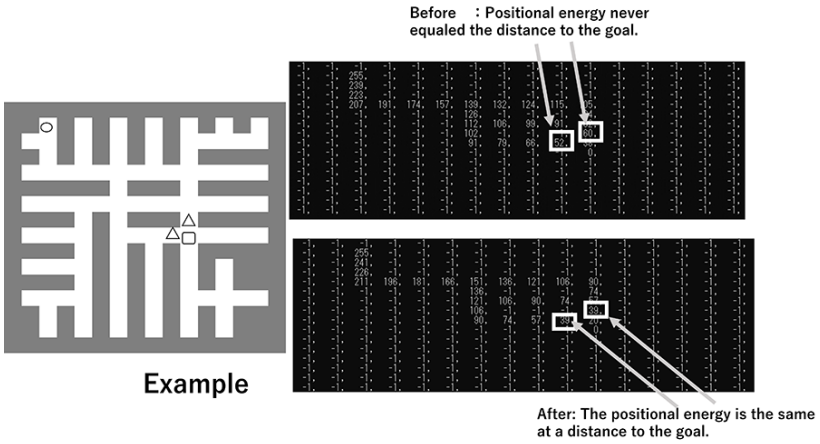

FIGURE 21. Improved evaluation of positional energy

TABLE 2. CPU performance

CPU	13thGen Intel(R) Core(TM)i9-13900K
Clock frequensy	5.8GHz
RAM	64GB (DDR5 4800MHz)
Primary cache	1.9MB
OS	Windows10 WSL2 (Ubuntu 20.04)
language	C
core	1core

$16 * 16$ square map of Figure 19, and the $32 * 16$ square map of Figure 18. The results confirmed that the search system using the $16 * 8$ square map of Figure 20 and the $16 * 16$ square map of Figure 19 is effective. The number of vertices in Dijkstra's algorithm is $W * H$ and the distance between adjacent vertices is 1. The FPGA used is Cyclone V 5CGXFC9E6F35C7 as shown in Table 4. We used 42 cores in a 16×8 square, 98 cores in a 16×16 square, and 210 cores in a 32×16 square in parallel processing. It does not calculate to the outer squares. The CPU in Table 2 is an Intel core i9 13900K. The maximum clocked CPU frequency is 5.8GHz. The language used was the C language. We converted NSL2vl to Verilog HDL in an environment using NSL (Next Synthesis Language). We converted as shown in Figure 22. The logic synthesis tool used was Intel Quartus Prime2 lite Edition. It is also a $32 * 16$ square as shown in Figure 23. The results of tests on mazes with various patterns showed that the method was 37.7 to 57.8 times faster for $16 * 8$ mazes and 46.6 to 75.8 times faster for $16 * 16$ mazes, and 56 to 156 times faster for $32 * 16$ mazes (Figure 23, Table 3). The Dijkstra algorithm includes data loading time, but not hardware. We compared cases of different map sizes. The larger the map, the more parallel operations and the faster the process. The larger the number of signed integer bits used for position energy, the larger the clock delay (Table 4). Multiple shortest paths allow for accurate positional energy evaluation (Figure 21).

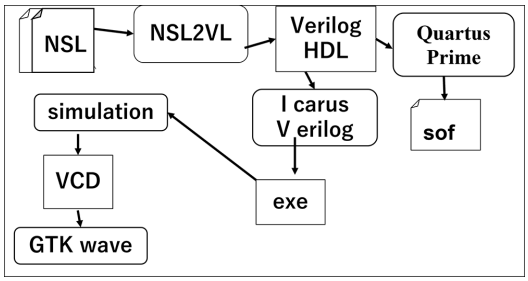

FIGURE 22. Development environment

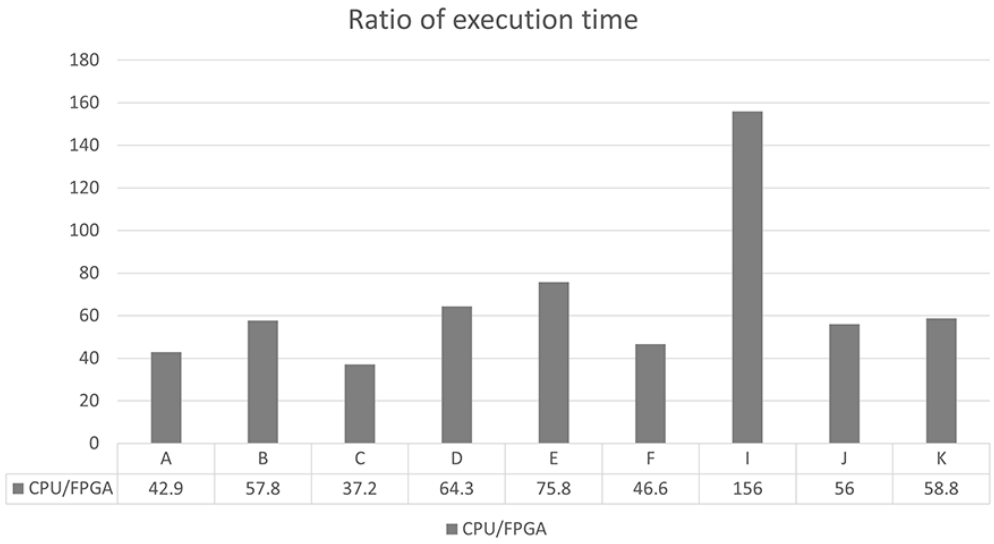

FIGURE 23. Performance

TABLE 3. FPGA CPU performance MI: Mitigation Method (FPGA), DI: Dijkstra Method, S: start, G: goal, M: MHz, G: GHz

	$16 * 8$					
	S = 33		S = 17			
	G = 65		G = 56		G = 109	
Time	MI 0.14 us	DI 6 us	MI 0.64 us	DI 37 us	MI 0.86 us	DI 32 us
DI/MI	A: 42.9		B: 57.8		C: 37.2	
FMAX	61.4 M	5.8 G	61.4 M	5.8 G	60.8 M	5.8 G
	$16 * 16$					
	S = 18		S = 17			
	G = 22		G = 151		G = 238	
Time	MI 0.14 us	DI 9 us	MI 1.7 us	DI 129 us	MI 1.8 us	DI 84 us
DI/MI	D: 64.3		E: 75.8		F: 46.6	
FMAX	56.1 M	5.8 G	55.8 M	5.8 G	46.2 M	5.8 G
	$32 * 16$					
	S = 34		S = 33			
	G = 38		G = 271		G = 477	
Time	MI 0.12 us	DI 20 us	MI 8.9 us	DI 499 us	MI 9.6 us	DI 564 us
DI/MI	I: 156		J: 56.0		K: 58.8	
FMAX	48.8 M	5.8 G	48.8 M	5.8 G	48.3 M	5.8 G

Table 4. FPGA performance S: start, G: goal

	$16*8$			$16*16$			$32*16$		
	A	B	C	D	E	F	I	J	K
S	33	17		18	17		34	33	
G	65	56	109	19	119	237	35	239	477
FPGA	Cyclone V 5CGXFC9E6F35C7								
Circuit scale (ALMs)	A: 9034/113,560 (8%) B: 10,210/113,560 (9%) C: 16,152/113,560 (9%)			D: 26,740/113,560 (24%) E: 26,640/113,560 (23%) F: 27,461/113,560 (24%)			I: 58,220/113,560 (51%) J: 58,132/113,560 (50%) K: 58,121/113,560 (50%)		
Number of integer bit of potential energy	8bit			9bit			10bit		
FPGA operationg clock perationg clock (clock)	8	31	51	8	96	100	6	432	466
Number of parallel processing cores	42 core			98 core			210 core		
Set up slack	−16.1 ns	−15.7 ns	−16.0 ns	−16.4 ns	−17.2 ns	−16.7 ns	−20.4 ns	−20.0 ns	−20.0 ns

6. **Conclusion.** We proposed a mitigation method to find the shortest path. Mitigation FPGA runs up to 156 times faster than Dijkstra for $32*16$ squares, up to 75.8 times faster for $16*16$ squares, and 57.8 times faster for $16*8$ squares. We have improved the accuracy of the evaluation of the positional energy of the mitigation method. Therefore, when multiple shortest paths exist, we have correctly evaluated all potential energies. A future issue is to compare this with the evaluation of positional energy by parallel processing in a single direction.

REFERENCES

[1] D. Wang, J. Feng, W. Zhou, X. Hao and X. Zhang, FCRoute: A fast FPGA connection router using soft routing space pruning algorithm, *Journal of Latex Class Files*, vol.14, no.8, 2021.

[2] E. W. Dijkstra, A note on two problems in connexion with graphs, *Numer. Math.*, vol.1, pp.269-271, DOI: 10.1007/BF01386390, 1959.

[3] S. Inoue and N. Shimizu, Acceleration of shortest path maze search, *Parthenon Society*, vol.48, pp.26-32, 2022.

[4] S. Lin, J. Liu, E. F. Y. Young and M. D. F. Wong, GAMER: GPU-accelerated maze routing, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol.42, no.2, pp.583-593, DOI: 10.1109/TCAD.2022.3184281, 2023.

[5] Y. Murata and Y. Mitani, A study of shortest path algorithms in maze images, *SICE Annual Conference*, Tokyo, Japan, pp.32-33, 2011.

[6] M. N. Sagming, R. Heymann and E. Hurwitz, Visualising and solving a maze using an artificial intelligence technique, *2019 IEEE AFRICON*, Accra, Ghana, pp.1-7, DOI: 10.1109/AFRICON46755.2019.9134044, 2019.

[7] C. J. Watkins and P. Dayan, Technical note: Q-learning, *Machine Learning*, vol.8, pp.279-292, DOI: 10.1023/A:1022676722315, 1992.

[8] D. Merrill, M. Garland and A. Grimshaw, Scalable GPU graph traversal, *ACM SIGPLAN Notices*, vol.47, no.8, pp.117-128, 2012.

[9] E. Yuriko, S. Ryuuichi and S. Masaaki, Consideration of dedicated hardware for pathfinding processing, *IPSJ SIG Technical Report, Research Report of Information Processing Society of Japan*, 2020-EMB-53, vol.19, pp.1-8, 2020.