最短経路迷路探索の高速化

東海大学　情報通信学研究科情報通信学専攻
井上　駿佑　清水尚彦
発表者 井上　駿佑

• はじめに

日々コンピューターの高性能化が求められる
迷路探索はインターネットのルーティングや自動運転の経路探索に役立つ
これまで逐次処理で最高速のダイクストラ法より速い並列処理の緩和法を NSL を用いてＦＰＧＡで開発した
(Shunsuke Inoue Naohiko Shimizu ” High-speed processing of shortest path maze search by miti- gation method Hardware with high-level action language NSL ” International Conference on Innovative Computing, Information and Control (ICICIC2022))
更に最短経路迷路探索の高速化を行った

迷路サイズ	32*16	16*16
従来手法処理時間/本手法処理時間	1.02	1.06
ダイクストラ法処理時間/本手法処理時間	26.4	56.0

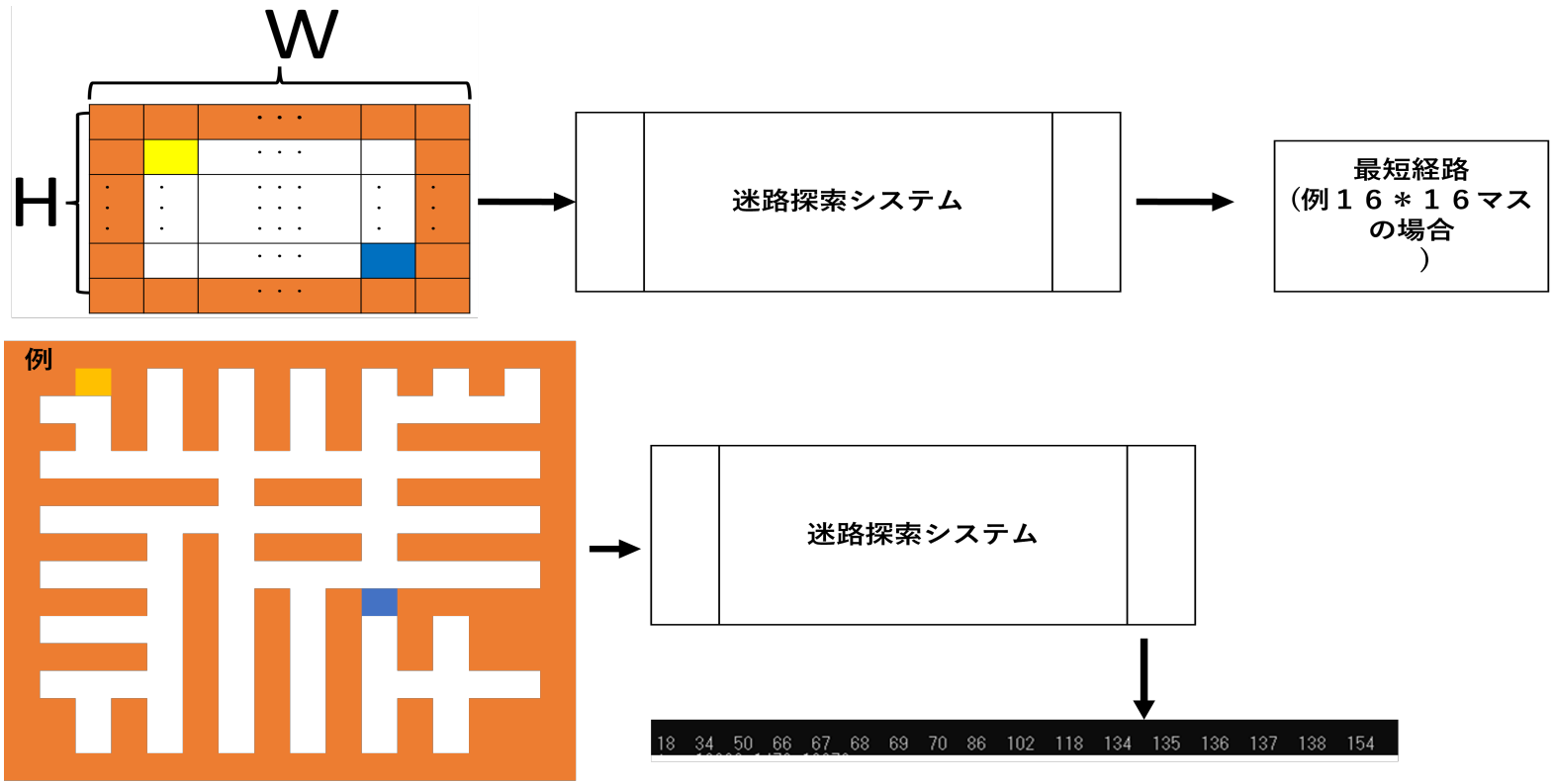
- 関連研究

GAMER 法で迷路の縦と横を各々並列処理でゴールからのユークリッド距離 (以後位置エネルギー) を計算し高速化した研究がある
(Shiju Lin, Jinwei Liu, Evangeline F.Y. Young, and Martin D.F. Wong, Fellow, IEEE , GAMER: GPU Accelerated Maze Routing IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 17 June 2022 DOI: 10.1109/TCAD.2022.3184281)
古川氏らは高解像度の画像復元のための行列の推定の反復法を並列処理により高速化の検証を行っている.
(古川拓実, 清水尚彦, ”スパースモデリングのハードウェア化による高速化”, 第 29 回 回路とシステムワークショップ, pp.225-230, 2016)

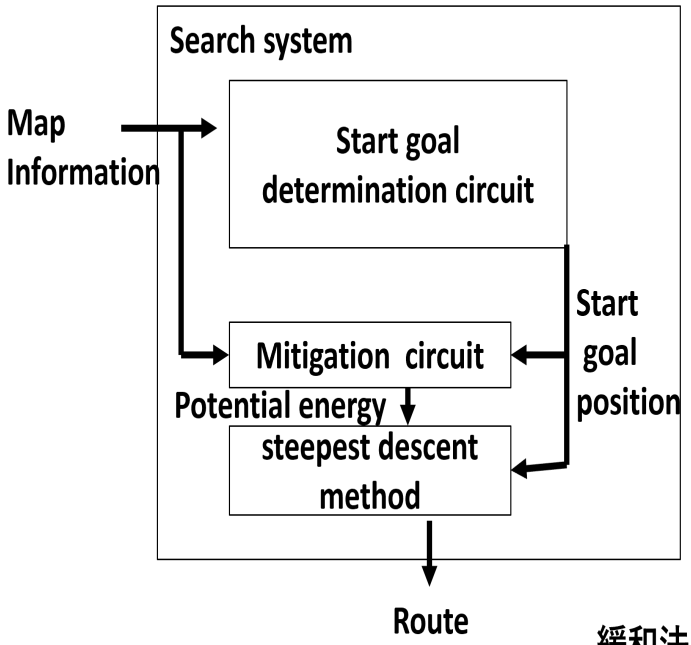
迷路探索

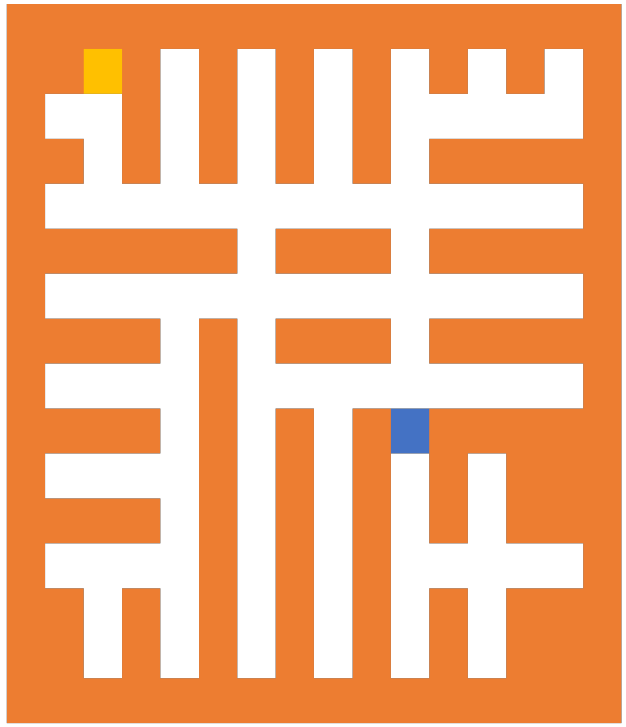
スタートとゴールの２点間の最短経路を求める

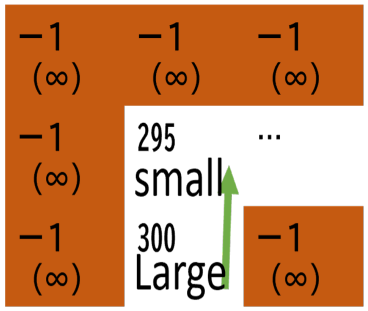
	Start	Goal	Load	Wall
Init Value	MAX	0	Middle	-1(∞)
Color	yellow	blue	white	brown



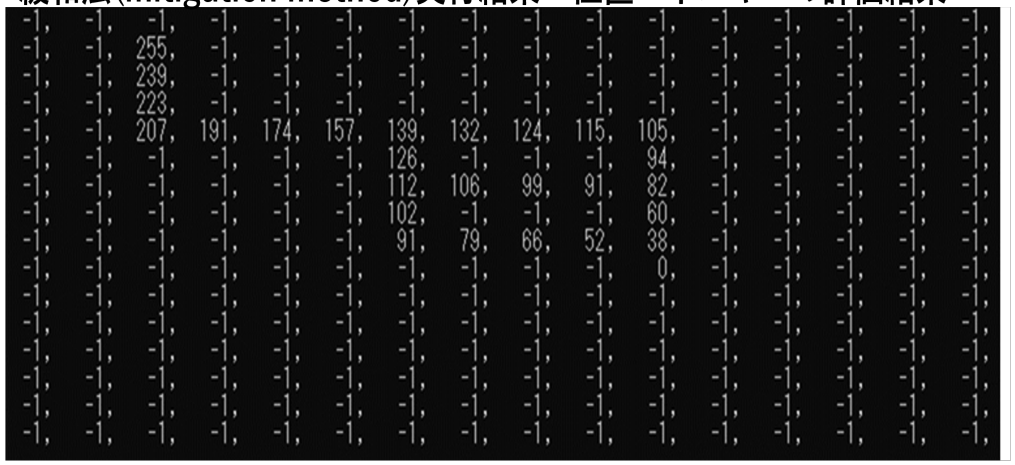
迷路探索システム





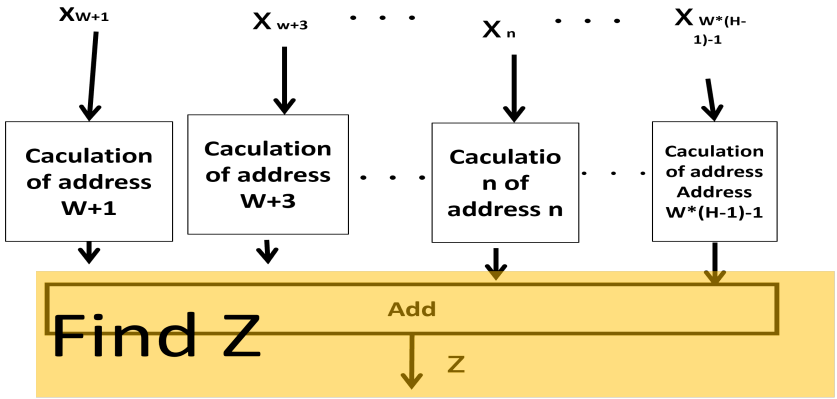


緩和法(mitigation method)実行結果　位置エネルギーの評価結果



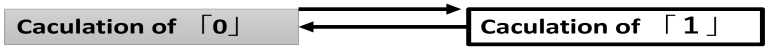
最急降下法(steepest method method)実行結果

緩和法の並列処理
　外枠は計算しないようにしている。
外枠は最急降下法に値を渡す時壁になるようにしている
ガウスザイデル法でマップの内側のマス数の半分のコアを使う。
(例: 横 32 ＊縦１６マス 210 コア,16*16 マス 98 コア)

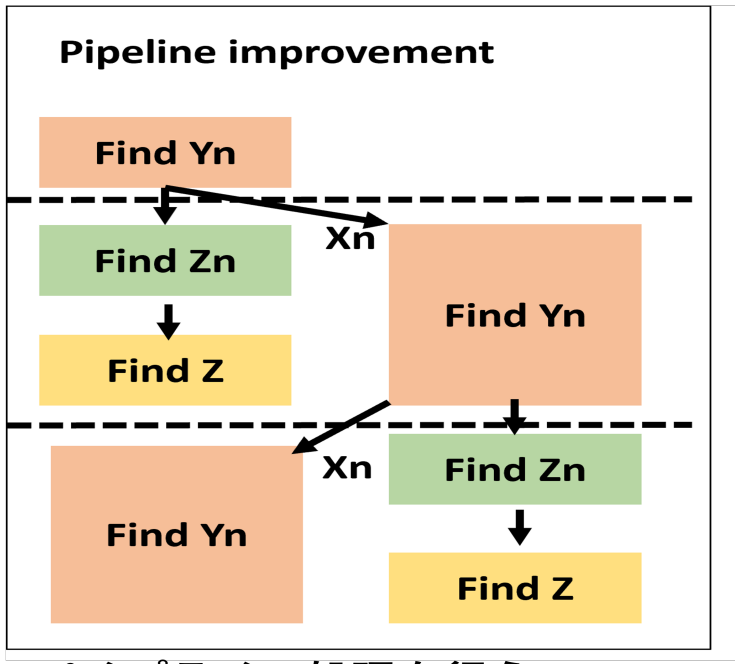


並列処理で各マスの計算を行う

WALL「0」	WALL「１」	WALL「0」	WALL「１」
WALL「１」	START「0」	LOAD「１」	WALL「0」
WALL「0」	LOAD「１」	WALL[0」	WALL「１」
WALL「１」	LOAD「0」	GOAL「１」	WALL「0」
WALL「0」	WALL「１」	WALL0」	WALL「１」



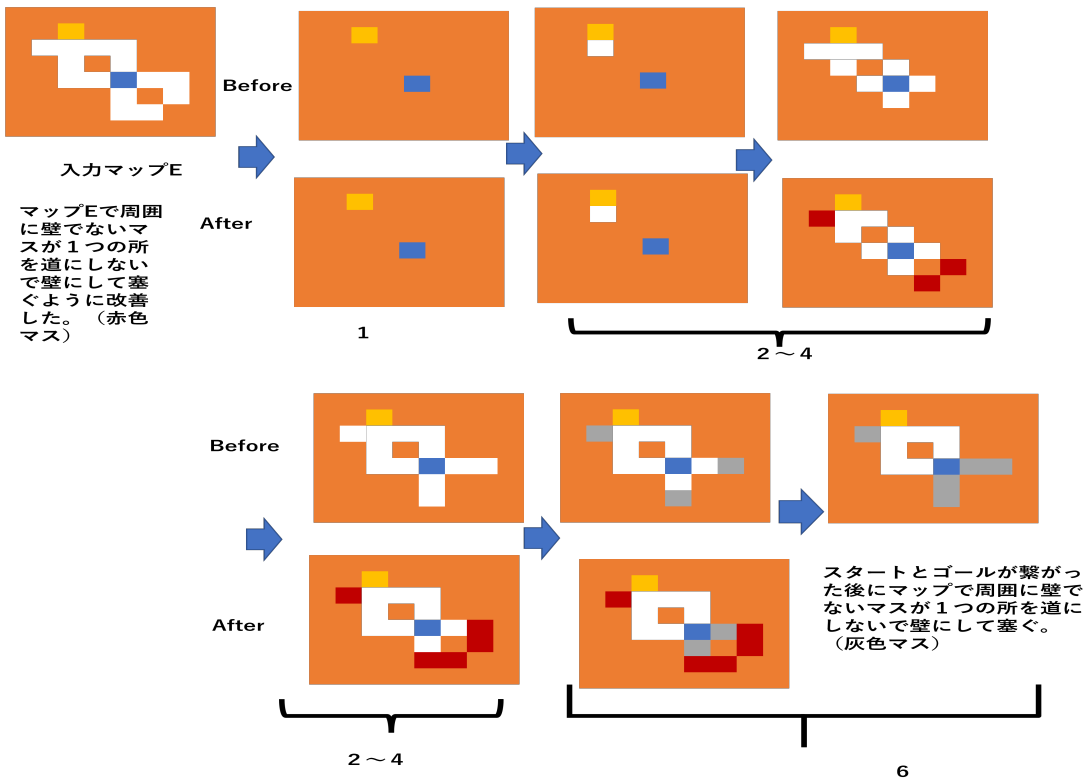
ガウスザイデル法で隣接する２マスを交互に計算



パイプライン処理を行う

緩和法フロー

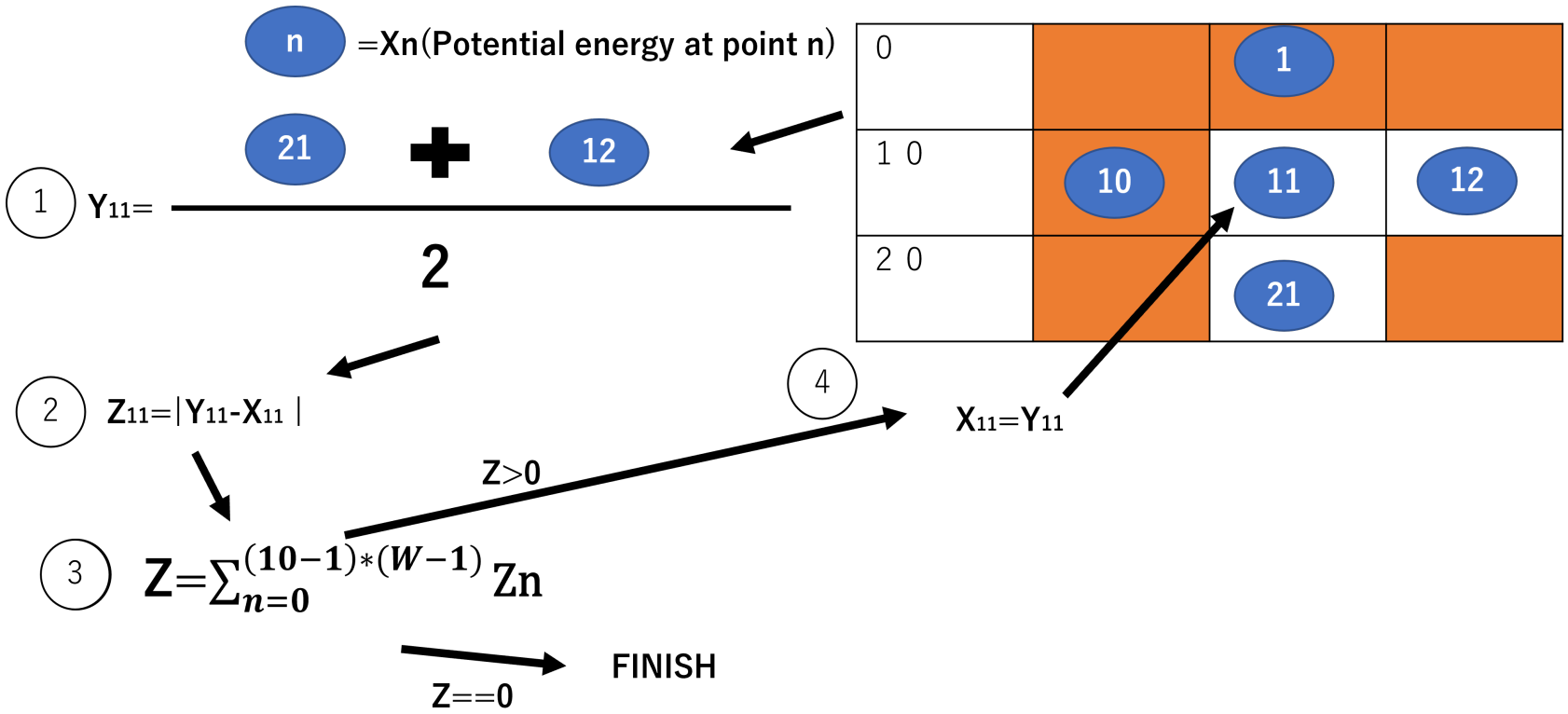
1. マップのスタートゴール以外を壁とした.
2. 入力されたマップの道である番地を見つける.
3. 見つけた番地の周囲が道でない番地が１方向である場合、その番地が壁から道にして道を掘る.
4. 道がスタートゴール双方から道が繋がったか確認する.
5. もし繋がったら次へ繋がらなければ２に戻る.
6. 道で周囲が１方向のみ壁でない場合道を壁にして塞ぐ.



緩和法位置エネルギー計算法 (道のマスの場合)

入力でマップの外枠の所は壁として代入する。

Figure: Caculution address n

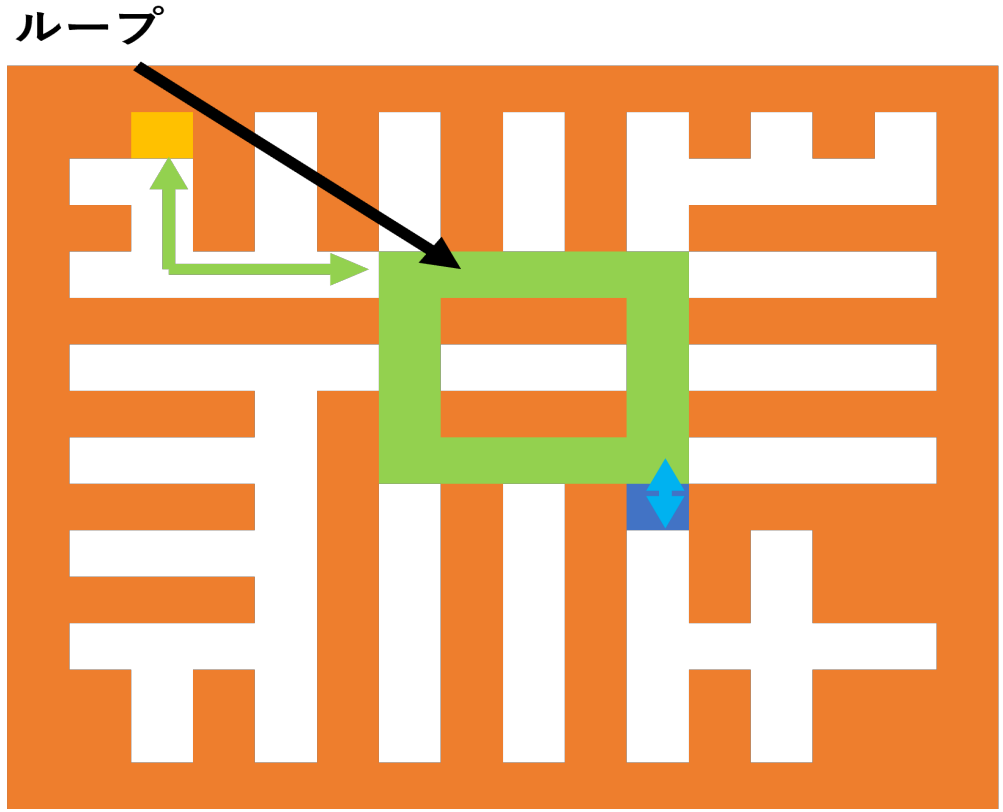


道以外のマスの位置エネルギーは計算せずそのままの値になる。

改良により期待できる点

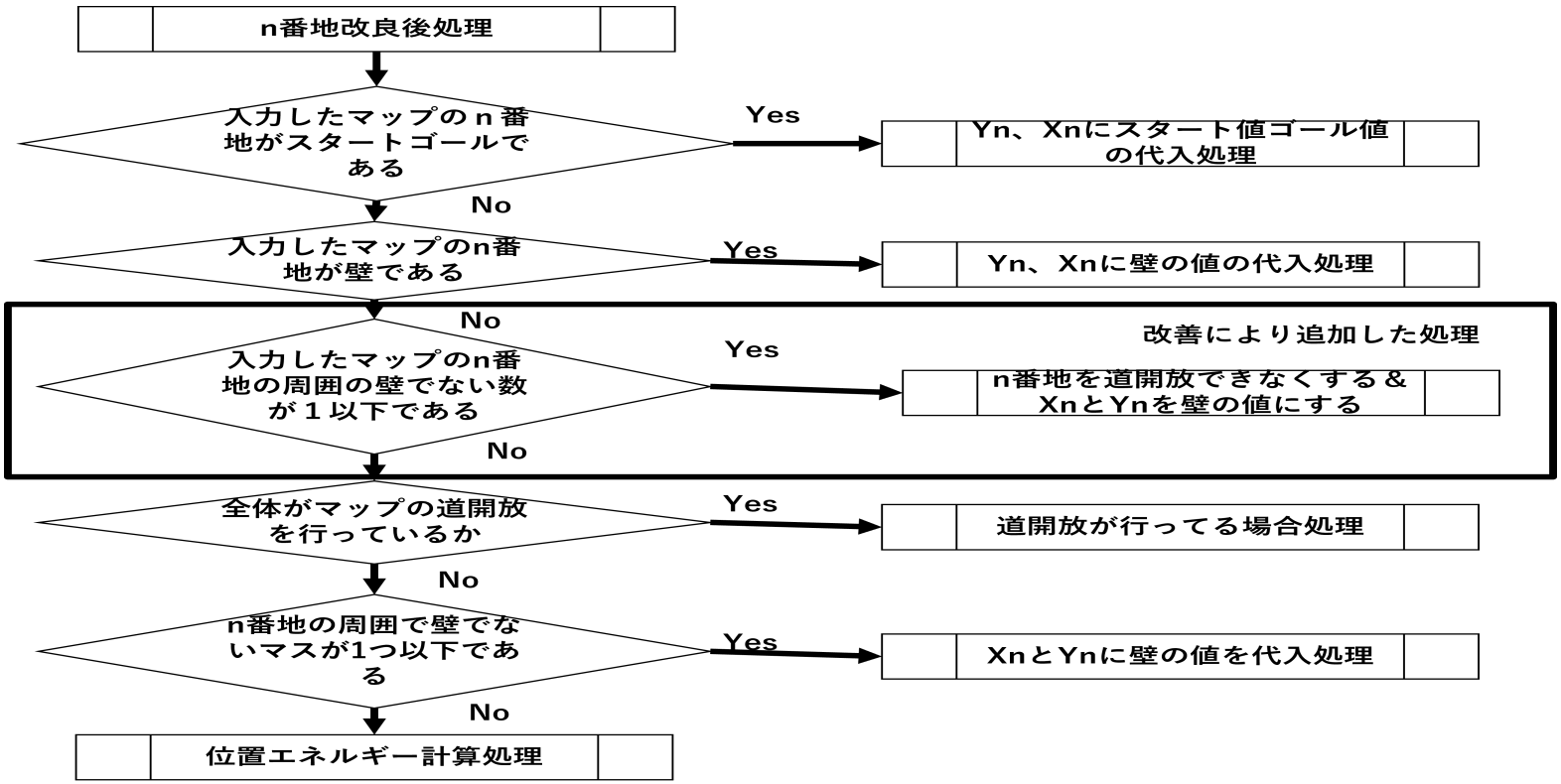
スタートゴールが入力したマップの両端にならなく、
掘れた道の番地が入力マップにループと袋小路があり、
ループがスタートゴールのどちらかに近い場合反復処理の回数が減ることで速くなる

スタートとループの距離＞
ゴールとループの距離

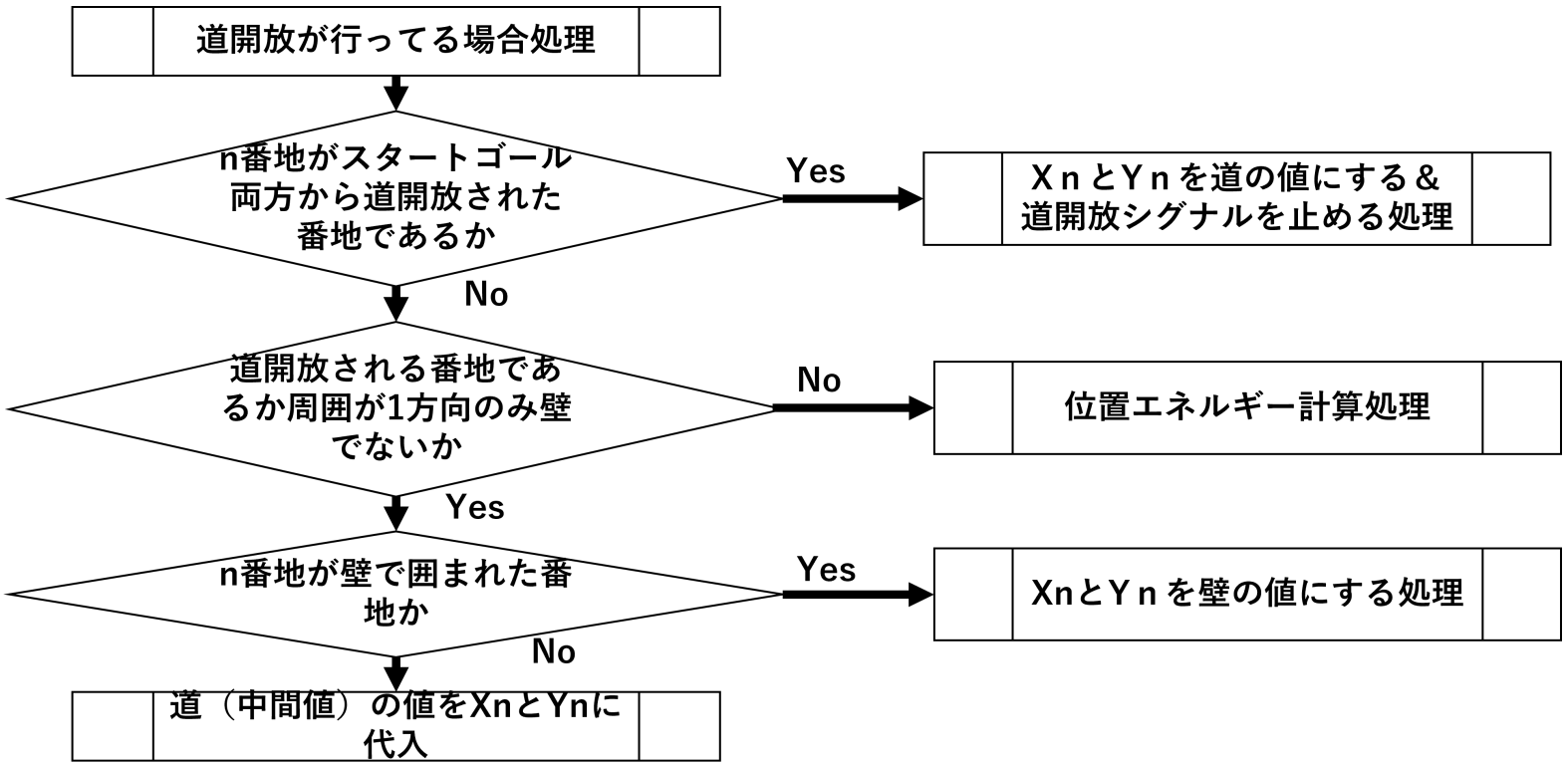


各マスの処理の改良点

各マスの処理で入力したマップＥの周囲のデータを扱った
入力したマップＥのｎマスが周囲が壁でないマスの個数の確認することを加えた。
壁でないマスの個数が１以下なら壁にマップＥを書き換える事を行った。

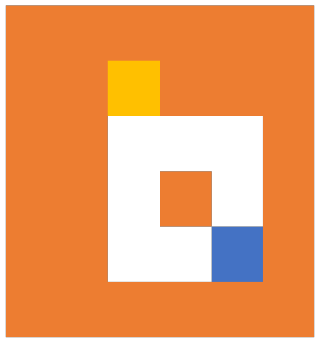


道開放の処理
スタートゴール双方から道開放されるマスができるまでこの処理は行う。
スタートゴールから道開放されたか管理するレジスタを使う。

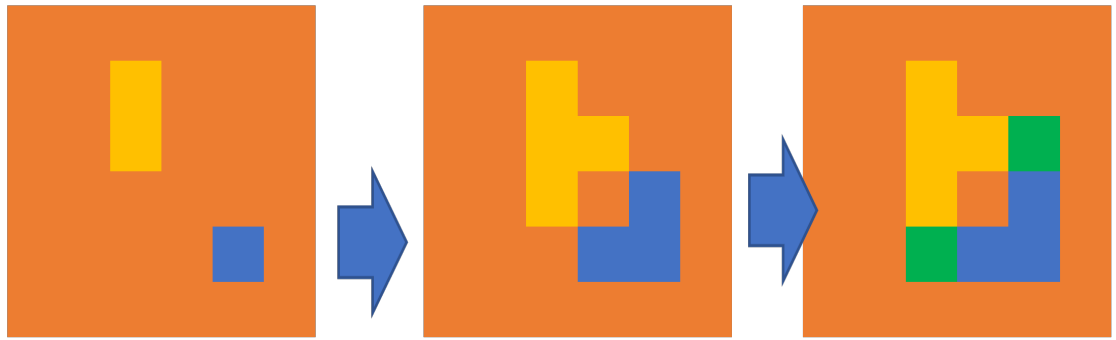


道開放の実現方法

道開放レジスタ：黄色が0b01青が0b10緑が0b11茶色が0ｂ00



入力したマップE



入力したマップEの道になっているｎマスで周囲4方向の道開放レジスタをOR演算の結果をｎマスの道開放レジスタに代入する。
0b11のマスが出たら道開放は終了となる

検証

CPU performance

CPU	13thGen Intel(R) Core(TM)i9-13900K
Clock frequensy	5.8 Ghz
RAM	64GB(DDR5 4533Mhz)
Primary cache	1.9 MB
OS	Windows10 WSL2 (Ubuntu 20.04)
language	C
core	1core

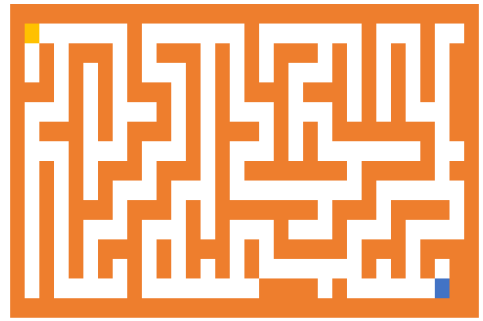


Figure: 32*16 maze

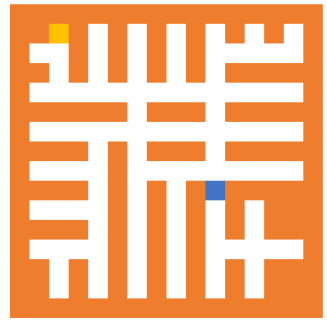


Figure: 16*16 maze

検証フロー
NSL よりソフトウエアシミュレータを生成し検証
緩和法回路の FMAX を論理合成ツール intel quatus prime2 で算出

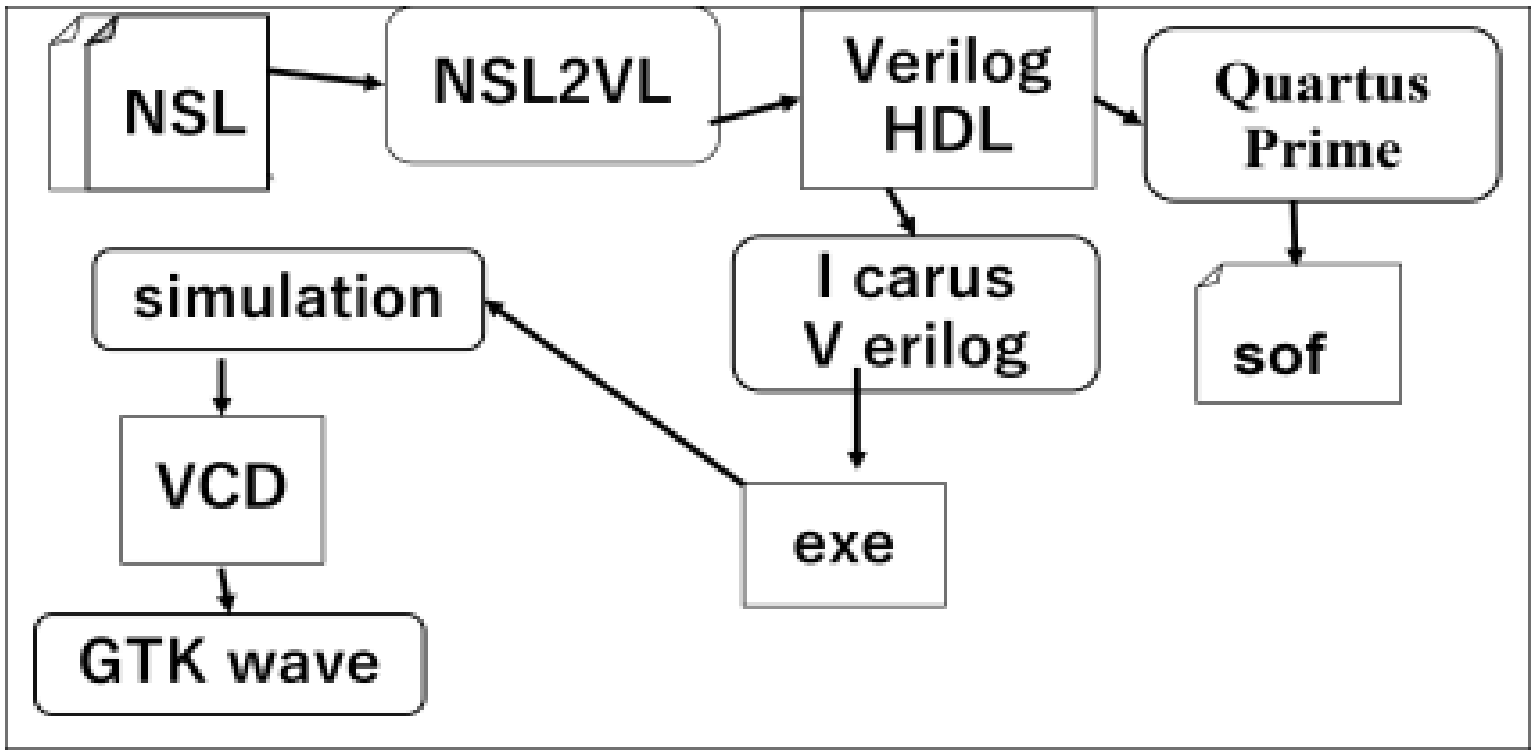


Table: FPGA CPU performance(A：以前の緩和法（FPGA）B: 改良後の緩和法（FPGA）C: ダイクストラ法（CPU）

	16*16			32*16		
	A	B	C	A	B	C
処理時間	1.6 us	1.5 us	84 us	14.2 us	13.8 us	365 us
FMAX	40 Mhz	40 Mhz	5.8 Ghz	31.9 Mhz	32.5 Mhz	5.8 Ghz
C/A	52.5			25.7		
C/B	56.0			26.4		
A/B	1.06			1.02		

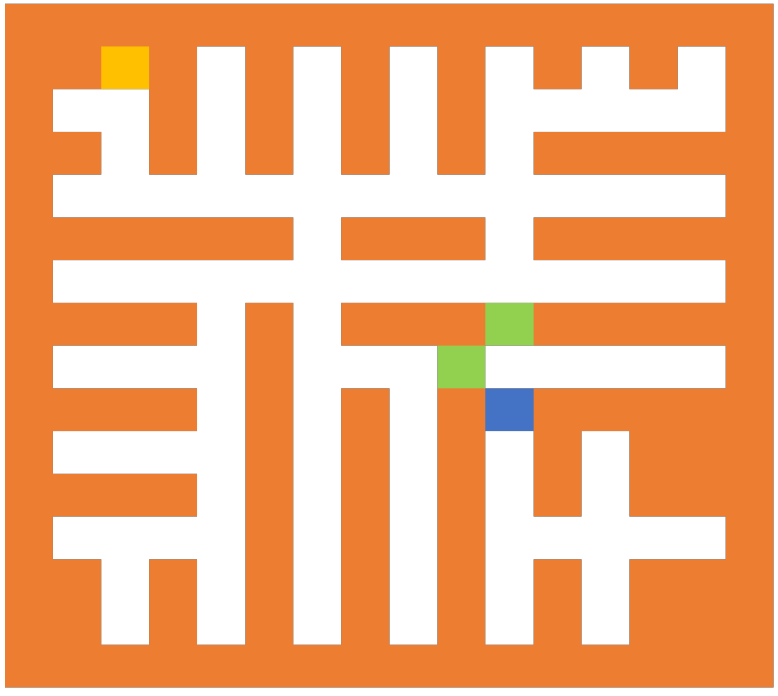
Table: FPGA performance

	32*16	
	before	after
FPGA	CycloneV 5CGXFC9E6F35C7	
Circuit scale(ALMs)	61,078/113,560(54%)	65,277/113,560(57%)
Number of integer bits of potential energy	10bit	
FPGA operationg clock (clock)	450	448
Number of parallel processing cores	210 core	

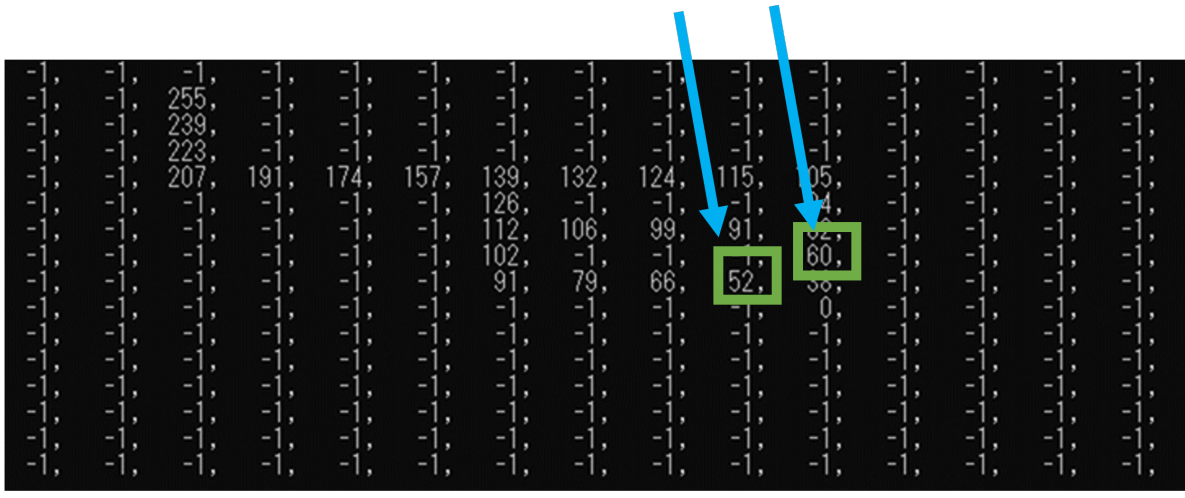
Table: FPGA performance

	16*16	
	before	after
FPGA	CycloneV 5CEFA7F31C6	
Circuit scale(ALMs)	26,097/56480(46%))	29,020/56480(51%)
Number of integer bits of potential energy	9bit	
FPGA operationg clock (clock)	64	60
Number of parallel processing cores	98 core	

改良すべき点
本来ならばゴールからの距離が同じである場合位置エネルギーも同じになる



問題点ゴールまでの距離で位置エネルギーが同じにならない



例

まとめ

緩和法の入力したマップ E の各マスの周囲 1 方向以下が壁でないマスを壁にして塞ぎ計算しないようにしてスタートゴールどちらかに近い方向にループが場合

16*16 マスで 6%32*16 マスで 2%高速化を行った

今後の課題

位置エネルギーの評価がゴールから同距離かつループしている位置で違う値になっている所の修正を行う