

歩行者向けデフォルメ地図生成のための 並列処理ハードウェアエンジンの設計

荒幡 明[†] 奈良 竜太[†] 戸川 望[†] 柳澤 政生[†] 大附 辰夫[†]

[†] 早稲田大学大学院基幹理工学研究科 情報学専攻
〒169-8555 東京都新宿区大久保 3-4-1
Tel: 03-3209-3211(5775), Fax: 03-3208-7439
E-mail: †arahata@togawa.cs.waseda.ac.jp

あらまし 現在携帯端末へ地図情報を配信するサービスが普及しているが、それらの地図の多くは PC 用の地図であり、微細な携帯端末用ディスプレイでの表示には適していない。地図情報は性質上リアルタイムの更新を必要とするため、あらかじめ視認性の高いデフォルメ化された地図を作成しておくのは現実的ではない。そのため地図情報を自動的にデフォルメ化する技術が多数開発・提案されているが、デフォルメ化処理をサーバ上で処理するとデフォルメ化地図をユーザの嗜好に合わせるのは困難である。携帯電話の処理能力ではデフォルメ化処理は処理量が大きく、現実的には実行不可能な程の処理時間を必要とする。本稿では携帯電話向けデフォルメ地図生成処理向け並列処理ハードウェアエンジンを提案する。携帯電話上でデフォルメ地図生成処理を可能とするために、処理量の削減に加えて、処理の並列化と並列処理ハードウェアエンジンを提案し、デフォルメ地図生成処理のボトルネック部分をハードウェア処理することで処理時間を短縮した。提案した並列処理ハードウェアエンジンによりデフォルメ地図生成処理は携帯電話上で1秒以内で処理出来る。

キーワード デフォルメ地図, 携帯電話, 並列処理, ハードウェア設計

A Parallel Hardware Engine for Generating Deformed Maps

Akira ARAHATA[†],

Ryuta NARA[†], Nozomu TOGAWA[†], Masao YANAGISAWA[†], and Tatsuo OHTSUKI[†]

[†] Dept. of Computer Science and Engineering, Waseda University
3-4-1 Okubo, Shinjuku, Tokyo 169-8555, Japan
Tel: +81-3-3209-3211(5775), Fax: +81-3-3208-7439
E-mail: †arahata@togawa.cs.waseda.ac.jp

Abstract Recently, many of the distribution of map information to mobile devices have been highly-popularized, however, those maps are generally for PC use and not suitable for displays as on mobile devices. According to the nature of map information, it has to be updated in real time, it is a distant idea to prepare an easy-to-read deformed map in advance. For that reason, it is difficult to tailor deformed map to preference of user when processing map on servers even automatic deformation of map data is proposed numerously. Mobile devices need loads of processing time which is virtually impossible in attribute to massive processing volume of data has to be required to deform map data by narrow throughput of mobile devices. In this paper, we propose parallel processing hardware engine for map deformation for mobile devices. We worked out to reduce processing time by processing on hardwares which was bottleneck of map deformation. Proposed parallel processing hardware engine can process deformation of map data within just 1 second on a mobile phone.

Key words deformed map, mobile phone, parallel processing, hardware design

1. ま え が き

今や日本における移動通信利用者は1億人を超え、第三世代携帯電話も契約者数が9千万人に迫っている。これによって今までよりも大きなデータの通信・処理が可能となり、テレビ電話などのマルチメディアサービスが普及し始めている。加えて携帯端末にGPS機能搭載が実現したことから、位置情報から地図情報を取得するアプリケーションが活発に考案、開発され、歩行者を対象とする携帯端末用地図情報配信システムが、研究、開発されている[10]。

地図情報はこれまでラスターデータで配信されていたが[4]、近年の携帯端末の高性能化により点の座標やそれを結ぶ曲線の描画情報であるベクタデータの動的な画面描画が可能となったため、ベクタデータでの配信が注目されている。ベクタデータはディスプレイの解像度に依存せず、拡大・縮小しても視認性が損なわれない利点がある。特に地図情報に関してはIEEEで定義されているSVGといわれるXML形式のベクタデータ形式が注目されており、今後その普及が広がっていくと考えられる[3]、[11]。

しかし携帯端末に配信される地図情報は元々PC用であるため、携帯端末上の微小なディスプレイにおいて見辛いものとなっている。そのため、現在携帯端末向けのデフォルメ地図を自動生成する技術が開発・提案されてきた[2]、[5]、[6]、[9]、[12]。デフォルメ化処理は地図の区画全体をデフォルメ化するもの[5]、[12]と、出発地点から目的地までの経路だけをデフォルメ化するもの[2]、[9]がある。その中でも文献[5]の手法では、研究段階に留まる他の手法と比べて、より実用的な道路形状デフォルメを提案し実用化へ向けて開発を進めている[6]。

地図情報そのものは更新されることが多く、リアルタイムにデフォルメ地図の作成が必要となる。これまで提案されてきた携帯端末向けのデフォルメ地図はサーバ上で地図情報をデフォルメ化処理したものを配信することを想定しているが、それではユーザの嗜好に合わせて頻りに地図情報を編集することが困難でありユーザビリティが低い。携帯電話向けデフォルメ化処理では、クライアントとなる携帯電話側でデフォルメ化処理を実現しユーザが望むデフォルメ化地図を自由に作成・表示することが望まれる。しかしながら既存の携帯電話上で、例えば文献[5]のデフォルメ化処理をするとは仮定すると、デフォルメ化処理は処理量が多く現実的に実行不可能な処理時間がかかる。携帯電話によるデフォルメ化処理を実現するには、デフォルメ化処理の処理量を削減するか、処理を高速化し、処理時間を縮める必要がある。

そこで本稿では、デフォルメ地図生成処理を2並列処理する携帯端末用デフォルメ地図生成ハードウェアエンジンを提案する。文献[5]のRNSモデルを用いたデフォルメ地図生成処理に携帯電話上での処理量を削減する2つの手法を提案する。更にボトルネックとなる処理を並列化し、設計した携帯端末用デフォルメ地図生成ハードウェアエンジンによる並列ハードウェア処理を提案する。本稿で提案した手法を用いることによって、現実的に実行不可能な処理時間を、1秒以内で処理できる。

本稿は以下のように構成される。2章では地図情報配信システムと、文献[5]のRNSモデルを用いたデフォルメ地図生成処理について紹介する。3章では携帯電話向けデフォルメ地図生成処理の問題点を明らかにした上で、問題点を解決するために処理量を削減する2つの手法と、並列処理ハードウェアエンジンを提案し、実装、評価する。

2. デフォルメ地図生成処理

自動でデフォルメ地図を生成する手法は数多く提案されているが、文献[5]のRNSモデルを用いた手法は他の手法が研究段階で留まる中で、実際に図1のように実用化に向けて開発が進んでいる[6]。本章ではデフォルメ地図生成処理について、デフォルメ地図生成処理サービスを提供する地図情報配信システム

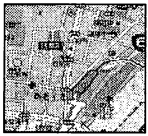
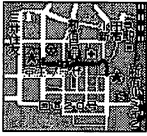
	オリジナルマップ	デフォルメマップ
浅草		
新宿		

図1 RNSモデルを用いた地図全体デフォルメ化処理例[6]。

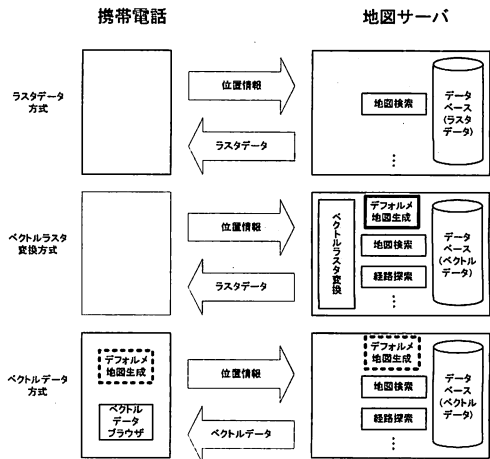


図2 地図情報配信システム概要図。

ムと、文献[5]のデフォルメ地図生成処理を紹介する。

2.1 地図情報配信システム[6]、[7]、[8]、[13]

地図情報配信システムはGPS付携帯電話やユーザ入力によって位置情報を送信し、地図情報を扱うアプリケーションサーバでデータベース(以降、地図サーバと呼ぶ)が受信し地図サーバ上のデータベースからその区画の地図情報を検索するシステムである。検索した地図情報を携帯電話に配信する上で3つの方式が存在する(図2)。

ラスターデータ方式 地図サーバのデータベース上に地図情報をラスターデータとして保存し、検索したラスターデータをそのまま送信する。サーバの処理量は少ないが、座標計算が複雑、地図のデータ量が膨大、決められた縮尺の地図情報しかない等の欠点も持つ。

ベクトルラスター変換方式 データベース上に地図情報をベクトルデータとして保存し、サーバがラスターデータに変換し携帯電話に送信する。座標計算や縮尺の指定が容易であるが、ベクトルラスター変換のサーバ処理負荷が高い。

ベクトルデータ方式 近年の携帯電話の高性能化によって実現した。データベース上に地図情報をベクトルデータとして保存し、携帯電話にベクトルデータを送信する。携帯電話側での視認性を損なわない縮小・拡大・回転、レイヤ合成が容易で、送信するデータ量も小さい。ベクトルデータの形式が統一化しておらず今後の標準化が期待されており、その中でSVG形式が最も有力とされている[4]。

現状ではベクトルラスター変換方式が最も普及しているが[7]、[8]、[13]、今後は視認性とデータ量が優れ携帯電話側での編集も容易なベクトルデータ方式に移行していくことが予想さ

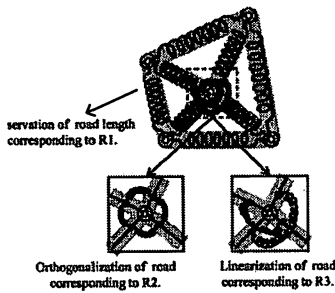


図3 RNSモデルの概要図[5].

れる[4]. ベクトルデータ方式では、データベース上のベクトルデータの地図情報は地理情報標準プロファイル(JPGIS)によって標準化されメタデータとして保持されている[3]. 地図サーバはこのメタデータから必要な地図情報を取捨選択し、経路探索を行い、必要があれば地図サーバが外部のコンテンツサーバにアクセスし更に情報を付け加える。このベクトルデータの地図情報がSVGなどに交換されて配信される。

デフォルメ地図生成処理は、現状では地図サーバ上でのデフォルメ処理が想定されている。地図情報配信システムは保持されている地図情報の書き換えが頻繁に起こるため、デフォルメ地図を予め作成することは出来ずリアルタイム処理しなければならない。デフォルメ処理をユーザの嗜好に合わせて頻繁に編集する場合、サーバ上でのデフォルメ地図生成は度々サーバにアクセスし地図情報を取得しなおすので、サーバ負荷の増大、携帯電話の通信処理の増大を招く問題を抱えている。本稿ではデフォルメ地図生成処理をユーザの携帯電話でのデフォルメ処理を想定する。携帯電話で処理する場合、一度地図情報を受信した後はユーザの嗜好に合わせて頻繁な編集が容易となるので、ユーザビリティの高いサービスを提供できる。

2.2 デフォルメ地図生成処理概要[5]

地図の区画全体をデフォルメ化するデフォルメ地図生成処理は、おおよそ以下のステップで順に処理される。

フィーチャセクション ユーザの嗜好性に基づいて地図上の必要なフィーチャ(地図を構成する道路・建物などのオブジェクト)を選択する。

形状デフォルメーション 選択したフィーチャの中から道路形状を見やすく変形する。

ランドマークモーフィング 変形した道路形状に合わせてランドマークを再配置する。

レイアウトコントロール 文字情報などを再配置する。

道路はノード(頂点)とエッジ(辺)からなるグラフ構造によってその形状が表現される。歩行者向けの地図情報は道案内を目的としているため、道路は2次元的なネットワーク構造を保持させたまま道路の形状をユーザにとって分かりやすくデフォルメ化することが重要であり、形状デフォルメーションは歩行者向けデフォルメ地図の分かりやすさを高める中心的な技術項目といえる。既存の研究では評価対象は評価用に作成された頂点数50程度の簡単な図形に限られていたが、文献[5]では道路形状正規化モデル(Road network Normalized Shaping model, 以下RNSモデル)を用いて100以上の頂点数からなる実際の地図を用いた形状デフォルメーションを実現させている。

2.3 RNSモデルを用いた最適化処理[5]

RNSモデルの概要を紹介する。RNSモデルは次に示す3つの生成規則をデフォルメ地図がもつ特徴として定義する。

R1 道路長を初期値のまま保存する。

R2 道路間のなす度を量子化する。

R3 道なりの道路を直線化する。

この生成規則を元に実際の地図情報がデフォルメ地図へと変形していくように定式化したものがRNSモデルで、道路ノード

の間にパネを埋め込んだモデルで表現される(図3)。ここである道路ノード j が持つパネエネルギーは、隣接する道路ノードを i, k として以下の式で表される。

$$E = \frac{1}{2} \sum_{i \rightarrow j} k_{ij}^v (ij \text{ 間長現在値} - ij \text{ 間長初期値})^2 - \sum_{i \rightarrow j \rightarrow k} k_{ijk}^o \cos(ijk \text{ 間角現在値} - ijk \text{ 間角理想値}) - \sum_{i \rightarrow j \rightarrow k} k_{ijk}^o \cos(ijk \text{ 間角現在値} - \pi) \quad (1)$$

$k_{ij}^v, k_{ijk}^o, k_{ijk}^o$ はパネ係数である。式の1項目が生成規則R1、2項目がR2、3項目がR3に対応し、パネエネルギーが小さければ小さいほど、生成規則に従ってデフォルメ化されている状態といえる。地図情報の各道路ノードのパネエネルギーの総和が最も小さいとき、その地図情報は最もデフォルメ化されている状態といえるので、各道路ノードをパネエネルギーが減少する方向へ移動させるため、ニュートン・ラフソン法の反復公式を適用し、各道路ノードの移動量を算出する。ここで各ノードの移動量を求めるとき、他のノードは移動しないと仮定し近似する準ニュートン法を適用している。最終的に各ノードの移動量は以下の式で与えられる。

$$\delta x_m^{(t)} = \frac{-b(y_m^{(t)}, y_m^{(t)})a(x_m^{(t)}) + b(x_m^{(t)}, y_m^{(t)})a(y_m^{(t)})}{b(x_m^{(t)}, x_m^{(t)})b(y_m^{(t)}, y_m^{(t)}) - b(x_m^{(t)}, y_m^{(t)})^2} \quad (2)$$

$$\delta y_m^{(t)} = \frac{b(y_m^{(t)}, x_m^{(t)})a(x_m^{(t)}) - b(x_m^{(t)}, x_m^{(t)})a(y_m^{(t)})}{b(x_m^{(t)}, x_m^{(t)})b(y_m^{(t)}, y_m^{(t)}) - b(x_m^{(t)}, y_m^{(t)})^2} \quad (3)$$

ただし $a(x_m^{(t)}) = \delta E / \delta x_m$, $a(y_m^{(t)}) = \delta E / \delta y_m$, $b(x_m^{(t)}, x_m^{(t)}) = \delta^2 E / \delta x_m \delta x_m$, $b(x_m^{(t)}, y_m^{(t)}) = \delta^2 E / \delta x_m \delta y_m$, $b(y_m^{(t)}, y_m^{(t)}) = \delta^2 E / \delta y_m \delta y_m$ である。地図情報に含まれる全ての各道路ノードに対して移動量を算出し、更新されていくノード配置が設定された収束条件を満たすまで繰り返す。

3. 携帯電話処理向けデフォルメ地図生成処理と並列処理ハードウェアエンジンの提案

RNSモデルを用いた形状デフォルメーションをサーバ側で、例えばPentium4 周波数3GHz、メモリ1GBで処理する場合、都心の500m四方地図サンプルのノード配置が収束するまでに20秒以上の時間がかかっている。地図情報配信システムはユーザのリアルタイムな地図情報の要求に応えるサービスなので、システムの一部であるデフォルメ地図生成もユーザがストレスを感じない1秒以下の短い時間で処理される必要がある。反復処理の初期段階でノード配置が急激に収束条件に近づいた後は徐々に収束条件に近づくことから、実際には収束する手前で反復処理を打ち切っても十分な効果が得られるが、それでも実際にサービスを商品として提供するには処理時間がかりすぎている。またサーバ側でデフォルメ地図生成処理する場合、ユーザの嗜好性に合わせてデフォルメ地図を自由に作成・表示することは困難である。

ここで携帯電話上でのデフォルメ地図生成処理を考える。携帯電話上でデフォルメ地図生成処理することによりユーザの嗜好性に合わせてデフォルメ地図を自由に作成・表示が可能となるが、携帯電話の処理能力ではサーバ上で処理する以上の処理時間がかかる可能性がある。携帯電話上でどの程度の処理時間がかかるか解析するため、RNSモデルを元に解析用のデフォルメ地図生成プログラムとして通常型プログラムを作成し、4つのサンプル図形を入力した。通常型プログラムを開発用のPCであるGenuine Intel(R) U1300 周波数1.06GHz メモリ1.49GBで処理し、出力された図形と処理時間を図4の太枠に示す。反復処理の繰り返し回数は収束されるまで繰り返されるが、本稿

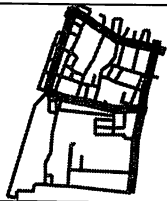
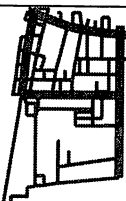
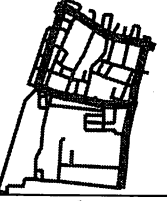
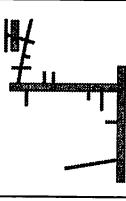
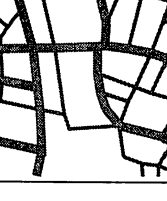
	入力図形	出力図形	計測条件	処理時間[s] (通常時を基準とした処理時間の比)	
				通常	固定小数点
高田馬場区画			ループ回数 2000 ノード数 126	通常	固定小数点
				2.43 (100%)	3.98 (164%)
				lock	混合
				1.35 (56%)	2.74 (113%)
高田馬場経路			ループ回数 2000 ノード数 38	通常	固定小数点
				0.50 (100%)	0.93 (186%)
				lock	混合
				0.32 (64%)	0.68 (136%)
高田馬場・新宿区画			ループ回数 2000 ノード数 97	通常	固定小数点
				0.90 (100%)	1.46 (166%)
				lock	混合
				0.58 (64%)	0.73 (81%)
西武新宿・新宿区画			ループ回数 2000 ノード数 56	通常	固定小数点
				0.94 (100%)	1.61 (171%)
				lock	混合
				0.67 (71%)	0.99 (105%)

図 4 デフォルメ地図生成処理のサンプル入出力図形と処理時間 (Genuine Intel(R) U1300 周波数 1.06GHz メモリ 1.49GB 上で動作させた場合)。

では比較のため一律 2000 回とした。通常型プログラムを携帯電話で動作させた場合を想定するため、ARM7TDMI 向けにクロスコンパイルし^(注1)、SimpleScalar^(注2)を用いてステップ数を見積もった結果を表 1 の通常型に示す。通常型プログラムに高田馬場区画のサンプル図形を入力した場合にかかるステップ数は、287,382M ステップで、仮に ARM7TDMI が 100MHz で処理したと仮定すると 2,837 秒必要であり、現実的には実行不可能である。

3.1 節では携帯電話上で現実的に実行可能な処理時間で処理出来るように、デフォルメ地図生成プログラム上で処理量を削減する手法を提案する。削減手法適用後のデフォルメ地図生成処理でも現実的には実行不可能な時間がかかってしまうので、3.2 節で携帯電話向け並列処理ハードウェアエンジンを提案しハードウェア処理させる。

3.1 デフォルメ地図生成処理の処理量削減手法

携帯電話上でデフォルメ地図生成処理の処理量を削減する lock 型プログラムと固定小数点型プログラム、そしてこの 2 つを組み合わせた混合型プログラムを提案する。この処理量削減手法を実装し、開発用の PC と ARM7TDMI 上で動作させた場合のステップ数を見積もる。

(注1)：クロスコンパイルには、gcc-3.3, binutils-2.14, gdb-5.3, newlib-1.11.0 を用いた。

(注2)：SimpleScalar は Winsconsin 大学によって開発されたシミュレータであり高性能なプロセッサの詳細なモデルを提供しており、シングルプロセッサシステムやプロセッサアーキテクチャの検証に広く利用されている。

表 1 SimpleScalar による ARM7TDMI 上で実行されたステップ数のシミュレーション結果 [M ステップ]。

	通常型	lock 型	固定小数点型	混合型
高田馬場区画 (比較)	287,382 (100%)	160,846 (56%)	113,707 (40%)	80,243 (28%)
高田馬場経路	61,600	—	—	19,365
馬場・新宿区画	102,112	—	—	20,097
西武新宿区画	113,762	—	—	28,132

3.1.1 lock 型

lock 型プログラムによる処理量削減手法を提案する。lock 型はデフォルメ地図生成処理の反復処理中において、移動量が 0 か極めて少ない収束したノードをその座標で固定 (lock) しそれ以降の反復処理で移動させない手法で、lock したノードの移動量算出分の処理を省くことが出来る (図 6(a))。RNS モデルは生成規則に従って反復処理しノード配置の最適解を求める手法であるが、厳密には求められるノード配置は局所解であり、時間をかけて最適解を求めるよりも近似解を素早く求めることを優先している。lock 型は処理を省くため出力されるノード配置を最適解から遠ざけてしまう可能性があるが、実際に実装したところ出力されたノード配置に目視で確認出来る差異は認められず実用上では問題ないと考えられる。実装した結果、全体の処理量のおおよそ 3 分の 1 の削減が可能となった (図 4, 表 1)。

3.1.2 固定小数点型

固定小数点型プログラムによる処理量削減手法を提案する。デフォルメ地図生成処理の移動量算出には浮動小数点演算が用いられているが、ARM7TDMI は浮動小数点用の命令を持たず、複合命令を用いて浮動小数点演算しているが処理時間の増

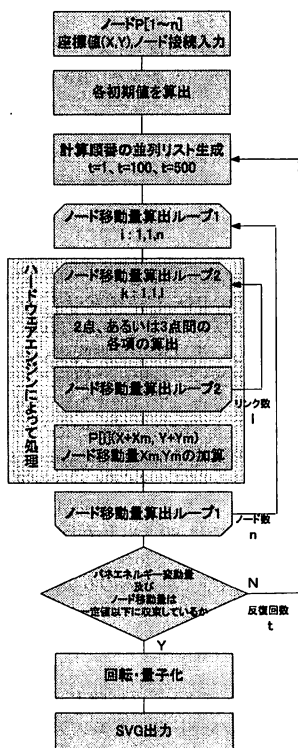


図 5 携帯電話処理向けデフォルメ地図生成処理プログラムチャート図。

大が考えられる。そこで浮動小数点で計算されている移動量算出処理を、固定小数点で実装した。小数部分の幅を 12 ビットとし、double 型の代わりに long long int 型で、また固定小数点用に arctan, sin, cos をテーブルで実装した。減衰関数として用いられる exp は 2 分の 1, 4 分の 1, 8 分の 1 を乗算することで代用する。ARM7TDMI は 32 ビットマシンであるが、long long int 型のような 64 ビットのデータを扱う命令を備えている。Genuine Intel(R) 上では処理時間が 1.6 倍程に増えたが(図 4)、SimpleScalar による ARM7TDMI 上でのシミュレーションではステップ数が 6 割近く削減できた(表 1)。

3.2 並列処理ハードウェアエンジンの設計

プログラムを改変した結果大幅にステップ数を削減出来たが、それでも混合型プログラムが高田馬場区画を処理するのに 80,243M ステップ必要で、100MHz で処理した場合 802 秒かかり、現実的に実行不可能である。処理時間短縮のため、デフォルメ地図生成処理のボトルネックをハードウェア処理する並列処理ハードウェアエンジンを提案する。デフォルメ地図生成処理の並列化には処理とハードウェア、双方の並列化が必要であるため、並列処理ハードウェアエンジンに合わせた計算順番の並列リスト生成を提案する。作成したデフォルメ地図生成処理に、提案する計算順番の並列リスト生成を加えたチャート図を図 5 に示す。RNS モデルを用いたデフォルメ地図生成処理におけるボトルネックは図 5 中四角で囲んだ繰り返し処理である。反復回数 t は本稿では実装比較のため一律 2000 回とする。

3.2.1 デフォルメ化処理並列化

デフォルメ地図生成処理を並列処理させる概念を図 6 に示す。ノードの処理順を A 列と B 列に並び替え、A 列と B 列に格納された順番に従って処理することで、ハードウェア並列処理を実現する。

RNS モデルの反復処理では、道路ノードを任意に与えられた番号に従って順番に移動量算出し、ノードの位置を更新する処理がなされる。地図情報中のノードはネットワーク構造を持

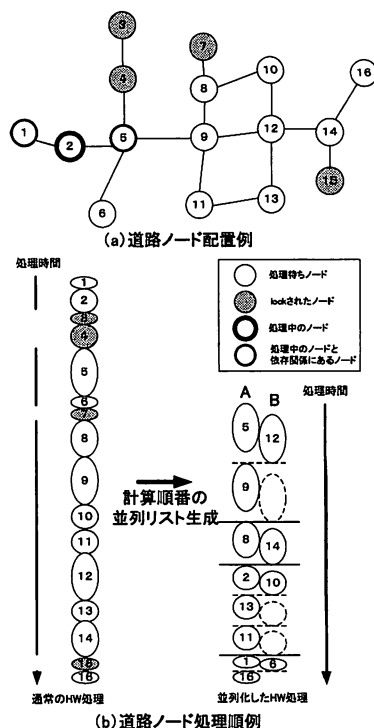


図 6 デフォルメ地図生成の並列処理概要図。

ち、デフォルメ化処理における各ノード算出時の移動量は、そのノードの座標とそのノードに隣接するノードの座標に依存する。例えば図 6(a) において、2 のノードを処理する時、1 と 5 のノードの座標値を参照して移動量を算出する。ここで 2 のノードと 1 のノードを並列に処理すると、2 が参照する 1 の座標値と 1 の移動量算出後の座標値とで差異が生じ、RNS モデルの生成規則に反する移動量を算出する可能性がある。そこで隣り合うノードが並列に処理されないように、ノードを計算する順番を図 6(b) の A 列と B 列のようになりリスト生成する。

計算順番のリスト生成プログラムの実装方法は、道路ノードに元々与えられている任意の番号を、A 列と B 列へ交互に格納していく。隣り合うノードの数(リンク数)によって移動量算出にかかる時間が違うので、リンク数が等しいノード同士を組み合わせる。A 列と B 列の同じ格納場所には、隣接したノードは隣り合わないようになら格納される。このとき最悪の場合(片側の列に全てのノードが格納される)が生じる可能性は、道路ノードの性質上極めて低く、現実的には A 列と B 列でほぼ均等な格納が可能であると考えられる。また lock 型と併用する場合、lock されたノードはリスト生成の対象から外す。

3.2.2 デフォルメ化処理ハードウェア並列化

ARM7TDMI コマンドセットは最大 16 個のコプロセッサを独立処理装置として接続し、特殊命令を実行する機能を追加することが可能である[1]。ARM7TDMI にハードウェアエンジンをコプロセッサに見立て、図 7 のようなシステムを想定する。ARM7TDMI はメモリからコプロセッサへ直接データの転送が可能である。提案する並列処理ハードウェアエンジンの構成を表 2 に示す。MAIN MODULE はバネエネルギーの偏微分値の計算、COMBINATION MODULE は偏微分値から移動量を計算し、レジスタにはノードの座標値や算出値が格納される。A 列を処理するハードウェア A(図 7 中の並列処理ハードウェアエンジン右半分)と B 列を処理するハードウェア B(左半分)の 2 組によって構成され、CONTROL UNIT によって制御される。

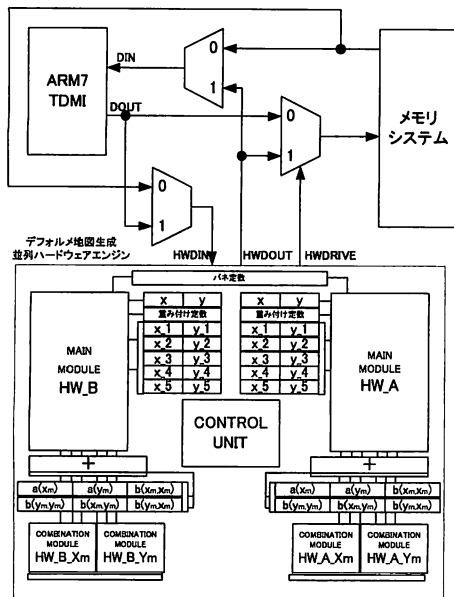


図 7 想定するデフォルメ地図生成処理システム全体図。

表 2 並列処理ハードウェアエンジンのモジュール構成表。

MAIN MODULE			
sqr テーブル	24bit*1024	×	1
arctan テーブル	24bit*1024	×	2
sin テーブル	24bit*1024	×	2
cos テーブル	24bit*1024	×	4
除算器	入力 24bit 出力 24bit	×	16
乗算器	入力 24bit 出力 24bit	×	40
その他	加算器、減算器など		
遅延 [ns]	65.7		

COMBINATION MODULE			
除算器	入力 24bit 出力 24bit	×	1
乗算器	入力 24bit 出力 24bit	×	4
その他	加算器、減算器など		
遅延 [ns]	36.1		

面積 [μm^2]			
MAIN MODULE	488,652	×	2
COMBINATION MODULE	38,037	×	4
CONTROL UNIT, etc	1,801	×	1
全体面積 [μm^2]	1,131,253		

ハードウェア A は順に以下大きく 3 つの状態に分けてノードの移動量計算を処理する。

1. データ入力 メモリから計算に必要なノード座標データとリンク先ノード座標データ、重み付け定数を計算処理の前に予め全て読み込む。
2. 計算 ノード座標データからノード移動量を計算する。
3. データ出力 計算した移動量を ARM7TDMI レジスタに出力する。

ハードウェア B も同様の状態に分かれて処理されるが、ハードウェア A とハードウェア B で各データ入力状態とデータ出力状態が競合しないように CONTROL UNIT によって制御される。ノードのリンク数が多くなるほど、データ入力と計算にかかるステップ数は多くなるので、ARM7TDMI からのリンク数に応じた命令によって制御を対応させる。本稿ではリンク数が最大で 5 まで対応できるように設計した。

3.2.3 実装結果

提案した並列処理ハードウェアエンジンをハードウェア記述言語 VerilogHDL で設計し、モジュール構成と Synopsys 社の DesignCompiler Z-2007.03-SP4 を用いて論理合成した場

表 3 高田馬場区画のデフォルメ地図生成処理をハードウェア処理した場合のステップ数見積もり値 [M ステップ]。

	HW 処理無 混合型	HW 単体処理 lock 型併用無	HW 並列処理 lock 型併用無	HW 並列処理 lock 型併用有
繰り返し処理 ステップ数理論値 (比較)	80,243	25.6 (100%)	18.5 (72%)	14.4 (56%)
その他処理ステップ数	12.9	12.9	13.1	13.5
合計ステップ数 (比較)	80,256	38.5 (100%)	31.6 (82%)	27.9 (72%)

合の遅延と面積を表 2 に示す。セルライブラリには STARC^(注3) (CMOS 90[nm]) の設計ルールを用い、制約なしで測定した。

ARM7TDMI を 100MHz で動作させたとすると、MAIN MODULE は 7 ステップで、COMBINATION MODULE は 4 ステップで処理できる。デフォルメ地図生成処理を提案する並列処理ハードウェアエンジンを用いて処理した場合のステップ数を表 3 に示す。ハードウェア処理する繰り返し部分と、メモリ・並列処理ハードウェア間の転送処理にかかるステップ数を理論値で、それ以外の処理を SimpleScalar により見積もった。繰り返し部分をハードウェア A のみで単体処理させることにより 0.26 秒、ハードウェア A と B で並列処理させることで 0.19 秒で処理できる。全体では 80,526 秒かかり現実的には実行不可能であった処理を、提案した lock 型と並列処理ハードウェアエンジンを用いることで、1 秒以内の 0.28 秒で処理でき、現実的に実行可能となった。

4. む す び

RNS モデルを用いたデフォルメ地図生成処理を解析し、処理量削減手法と並列処理ハードウェアエンジンを提案した。提案した並列処理ハードウェアエンジンを組み込むことで、デフォルメ地図生成処理を 1 秒以内で処理可能である。携帯電話上で実際に動作させることが可能かどうか、より正確に見積もるためにシステム全体を構築し、消費電力なども含めて今後考察する予定である。

文 献

- [1] ARM7TDMI テクニカルリファレンスマニュアル, ARM 社ホームページ, <http://www.jp.arm.com/>
- [2] 馬場口登, 堀江政彦, 上田俊弘, 淡誠一郎, 北橋宏宏, “経路理解支援のための略地図とその案内文の生成システム,” 電子情報通信学会論文誌, vol.J80-D-2 no.3, pp.791-800, 1997.
- [3] 国土交通省ホームページ, <http://www.mlit.go.jp/>
- [4] 小林重令, “The graphics information sharing platform for mobile computing based on SVG,” KDDI 研究所, 2005.
- [5] 丸山貴志子, 谷崎正明, 嶋田茂, “デフォルメマップ生成のための道路形状正規化モデル,” 電子情報通信学会論文誌, vol.J87-A no.1, pp.108-109, 2004.
- [6] 丸山貴志子, 谷崎正明, 嶋田茂, 伏木匠, “交通情報のデフォルメ地図へのマッピング方式とモバイル交通情報提供システムへの適用,” <http://www.ipsj.or.jp/10jigyo/fit/fit2004/fit2004program/html/event/pdf/17maruyama.pdf>, 2004.
- [7] MapFan ホームページ, <http://www.mapfan.com/>
- [8] NAVITIME ホームページ, <http://www.navitime.co.jp/>
- [9] 二宮直也, “歩行者ナビゲーションにおける微小画面での視認性とユーザの迷いにくさを考慮した略地図生成手法,” 電子情報通信学会技術研究報告, ITS2006-34, 2006.
- [10] 総務省ホームページ, “情報通信白書平成 19 年版,” <http://www.soumu.go.jp/>
- [11] W3C, <http://www.w3.org/Graphics/SVG/>
- [12] 山守一徳, 本田宏, 長谷川純一, “ストリート単位の変形に基づく道路網の整形手法,” 情報処理学会論文誌, vol.J84-D2, no.9, pp.2058-2069, 2001.
- [13] ZENRIN ホームページ, <http://www.zenrin.co.jp/>

(注3) : STARC[90nm] ライブラリは東京大学大規模集積システム設計教育研究センターを通し、株式会社半導体理工学研究センター (STARC) と株式会社先端 SoC 基盤技術開発 (ASPLA) の協力で開催されたものである。