3J-05

高位合成によるブロック LU 分解のハードウェア化

前田康平† 瀬戸 謙修†

東京都市大学 工学部 電気電子工学科†

1 はじめに

工学分野のシミュレーションで出現する大規模な連立一次方程式の求解を高速化するためにLU分解が使用されることが多い。LU分解は行列を二つの三角行列の積の形に分解する手法であり、逆行列を用いて連立一次方程式を解く場合に比べ高速に解を得ることができる。LU分解を高速化することがシミュレーション時間の短縮につながるためLU分解の高速化について様々な研究がなされている[1]。本研究では、LU分解を高速化するために専用ハードウェア化して高速化を図り、時間の短縮を行う。その際に有効な手法であるブロックLU分解を動作記述からハードウェアに合成する高位合成を用いて高速化を図る。

2 ブロック LU 分解

LU分解をハードウェア化する際に有効な手法であるブロックLU分解[2]は元の行列をブロックに分割し、各ブロックに演算を並列実行することで高速化する手法である。図１のように元の行列Ａを２×２に分割する場合について考える。ここでNは元の行列サイズ、Ｂは分割後のA11部分の行列サイズである。

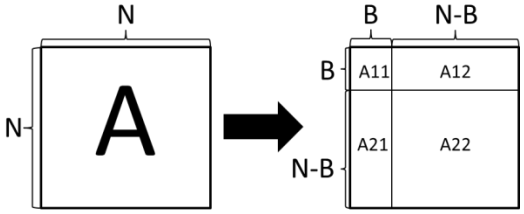

図１：行列のブロック分割(2×2)

また、分割後のL、Uは図2のように表される。

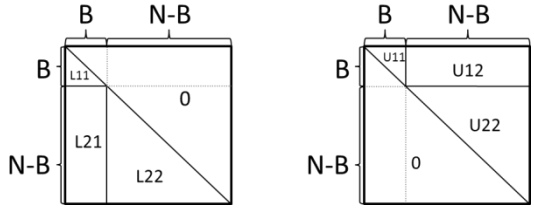

図2：分割後のL,U

分割後のL12,U21は値が0なので不要となる。

図３にブロックLU分解のアルゴリズムを示す。ブロックLU分解は分割後のA11部分に通常のLU分解を行いL11、U11を求める。得られたL11、U11の逆行列を計算しA12からU12、A21からL21を求める。得られたL21、U21の積をA22から引き、求まる新たなA22をA22'とする。このA22'を新たな行列Aとし、再度分割して計算を繰り返す。

Hardware Design of Block LU Decomposition with high-level synthesis

Maeda Kohei† and Seto Kenshu†

†Tokyo City University Faculty of Engineering Electrical and Electronic Engineering

これらの計算を繰り返して、三角行列LとUを求める。

図３のアルゴリズムの内で演算はステップ4,5,6で行うためこれらの演算を並列実行して高速化を図る。

アルゴリズム ブロックLU分解
入力：A n×nの行列
出力：三角行列L、U
1: while Aの行数がBの行数より多い do
2: Aを4つに分割（A_{11},A_{12},A_{21},A_{22}）
3: A_{11}にLU分解を行いL_{11},U_{11}を得る
4: $L_{11}^{-1}\times A_{12}$からU_{12}を得る
5: $A_{21}\times U_{11}^{-1}$からL_{21}を得る
6: $A'_{22}=A_{22}-(L_{21}\times U_{12})$
7: A_{22}を新たなAとする

図３：ブロックLU分解のアルゴリズム

実際に高位合成でブロックLU分解の並列実行を行う場合は分割数を増やし、より並列演算を行うことが必要である。例として、図４のように元の行列を3×3に分割した場合のブロックLU分解について考える。図4のブロック分割は図3のアルゴリズムのステップ２でAを9個のブロックに分割したものである。ここで、図４のMは任意の値であるが今回は並列演算を行うために(N-B)/2とした。アルゴリズムにおけるU、Lを求めるための演算は式１のように表される。

$$U_{12}=L_{11}^{-1}\times A_{12}\ ,\ U_{13}=L_{11}^{-1}\times A_{13}$$
$$L_{21}=A_{21}\times U_{11}^{-1}\ ,\ L_{31}=A_{31}\times U_{11}^{-1} \quad (1)$$

このとき U12,U13,L21,L31 に対して並列に演算を行うことで通常の LU 分解に比べ高速な演算を可能にすることがブロック LU 分解の利点である。残りの A22～A33 の演算については式２のように表される。

$$A_{22}=L_{21}\times U_{12}\ ,\ A_{23}=L_{21}\times U_{13}$$
$$A_{32}=L_{31}\times U_{12}\ ,\ A_{33}=L_{31}\times U_{13} \quad (2)$$

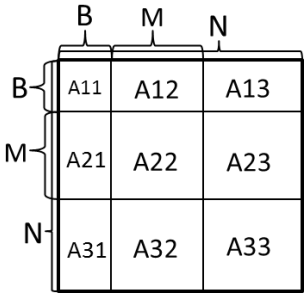

図４：行列のブロック分割（３×３）

演算後の A22～A33 を新たな A 行列として再度分割し、演算を繰り返すことで LU 分解を行う。

3 提案手法

ブロック LU 分解の高速化には並列実行が重要である。そのため、図５のように L,U に相当する A12,A13,A21,A31 をさらに半分に分割して演算を行うことを提案する。図５のように分割することで U を求める部分は式３のように

表せる。

$$U_{12}=L_{11}^{-1}\times A_{12}\ ,U_{13}=L_{11}^{-1}\times A_{13}$$
$$U'_{12}=L_{11}^{-1}\times A'_{12}\ ,\ U'_{13}=L_{11}^{-1}\times A'_{13} \quad (3)$$

式３よりアルゴリズムのステップ４,５で行う L,U を求める演算は２倍の並列演算が可能となり、アルゴリズムのステップ６で行う A の演算については L,U がそれぞれ２倍となるため４倍の並列演算が可能になる。

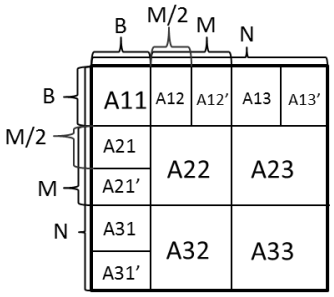

図５：提案手法による分割（３×３）

なお、図５のように半分に分割した結果メモリの word 数の総数は変わらないがメモリ自体の数が増えるため bit 幅の総数は増えるので３分割以上の分割はここでは行わないとする。

4　実験

今回、ブロックの分割数、行列のサイズを変えてブロック LU 分解ハードウェアの合成を行った。高位合成は vivado_hls で行い、ターゲット FPGA は vertex-7 で、動作周波数は 100MHz とした。最も内側のループは全てループパイプライン化による並列で行った。B のサイズは N の５分の１とした。表１は分割数を 3×3 から 5×5 まで変化させたときの合成結果を示しており N は行列サイズ、Latency は分解にかかったサイクル数(kcycle)、速度向上率は従来のブロック LU 分解に比べ提案手法がどれだけ性能向上したかを示している。図６は分割数毎の Latency をグラフに示している。

表１：ブロック LU 分解合成結果（分割数）

	分割数	N	Latency	速度向上率
従来のブロック LU 分解	3×3	50	298	1.00
提案手法	3×3	50	148	2.02
従来のブロック LU 分解	4×4	50	201	1.00
提案手法	4×4	50	114	1.75
従来のブロック LU 分解	5×5	50	148	1.00
提案手法	5×5	50	95	1.55

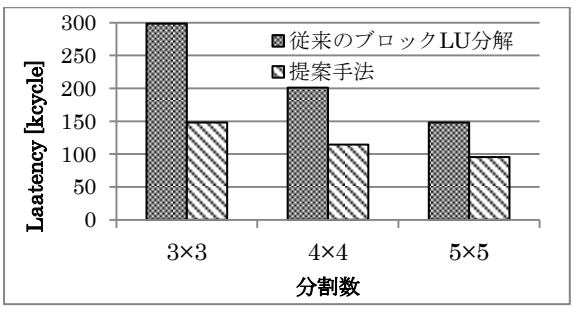

図６：分割数毎の速度比較

図６より分割数を増やすほど提案手法による性能向上効果が低くなっていることがわかる。行列のサイズが小さい場合に分割数を増やすと演算にかかる時間は短縮されるがメモリの読み込み・書き込みにかかる総時間は変わらない。そのため全体の時間に対する演算時間の割合が減り、提案手法によって短縮できる時間自体が減少する。このことから表１のような結果となる。

次に分割数 5×5 のときの行列サイズ N を変化させた場合についてブロック LU 分解を行った結果を表２に示す。

表２：ブロック LU 分解合成結果（行列サイズ）

	N	Latency	速度向上率
従来のブロック LU 分解	25	23	1.00
提案手法	25	18	1.30
従来のブロックLU 分解	50	147	1.00
提案手法	50	95	1.55
従来のブロックLU 分解	75	320	1.00
提案手法	75	181	1.76
従来のブロックLU 分解	100	775	1.00
提案手法	100	414	1.87

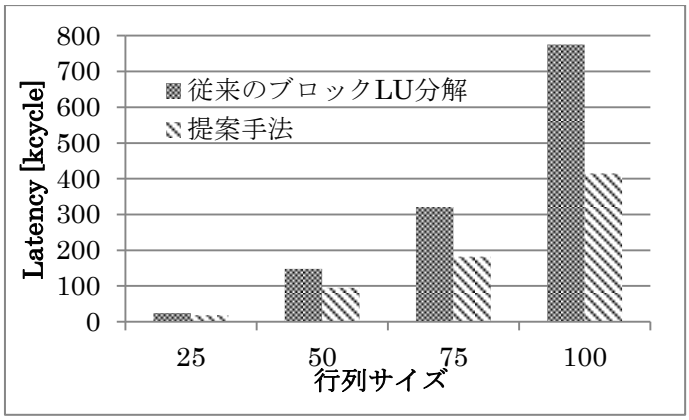

図 7：行列サイズ毎の速度比較

表２より行列サイズが大きいほど提案手法の効果が表れることがわかる。

5　結論

本稿では、高位合成によりブロック LU 分解のハードウェア化を行った。従来のブロック LU 分解アルゴリズムに対し部分的に分割数を変えて並列演算量を増やす手法を提案した。提案手法を適用した結果、従来のブロック LU 分解のアルゴリズムから高位合成した場合に比べ最大で 2.02 倍性能向上できた。

参考文献

[1] Manish Kumar Jaiswal and Nitin Chandrachoodan “FPGA-Based High-Performance and Scalable Block LU Decomposition Architecture” IEEE TRANSACTIONS ON COMPUTERS, VOL. 61, NO. 1, JANUARY 2012

[2] L. Zhuo and V.K. Prasanna, “High-Performance and Parameterized Matrix Factorization on FPGAs,” Proc. Int'l Conf. Field Programmable Logic and Applications (FPL '06), pp. 1-6, Aug. 2006.