2022 年 2 月 8 日

令和 3 年度 情報・ネットワーク工学専攻 コンピュータサイエンスプログラム 修士研究発表会

FPGA を用いた重力多体問題の 高位合成に関する最適パラメータ推定とツール間性能比較

コンピュータ・サイエンスプログラム 学籍番号:2031140 成見研究室 村松耀生

1 はじめに

複数の計算コアを有する GPU を用いた数値計算アクセラレーションが脚光を浴びているが，消費電力の増大が懸念されている．それに対して，再構成可能な論理デバイスである FPGA が低消費電力性と計算高速性の両立が可能という理由から近年，注目を集めている．

ただし，FPGA を含む大規模集積回路設計開発は，回路仕様の設計や実装，デバッグや最適化など様々な開発に大きなコストを要していた．そこで，FPGA の大規模化に伴い，C 言語等の高級言語で設計する高位合成が使われ始めている．Xilinx 社は高位合成ツールとして 2015 年に SDSoC[1] を発表し，2019 年にはその後継となる Vitis[2] を発表した．

しかし，高位合成による記述は RTL より抽象度が高いため,RTL の記述に比較して記述量が少なく設計柔軟性・生産性共に優れる反面，様々な論理資源への割り振りは高位合成ツールに委ねられる．

そこで，本研究では Vitis を用いた高位合成において最大性能を出すためのパラメータを推定する．その前段階で，まずは SDSoC と Vitis の違いについて調査する．高位合成で性能を出すためには様々な最適化オプションを適切に使用する必要があるが，ツールによってその挙動は変わるためである．次に性能モデルを構築し最適パラメータを推定する

2 既存研究

Jinguji らは SDSoC を用いて機械学習の一つであるランダムフォーレストを FPGA に実装し，CPU や GPU よりも絶対性能や電力性能がよいことを示した [3].Skalicky らは画像処理に関して SDSoC を経由することで Python から高位合成するツールを開発し，CPU で最適化された OpenCV のルーチンよりも高速に処理できることを示した [4]. 近年はエッジデバイスで AI 関連の処理を行わせるニーズが高まっているが，Kalantar ら [5] や Wang ら [6] は Vitis の AI 向けのツールである VitisAI を用いて CNN を加速した．

高位合成ツールを用いてこのようなアクセラレータを開発する際には，性能を予測することが課題となる．これはツールが高度であるためにどのような最適化がなされどの程度の性能が出るかが単純には予測出来ないためである．Shahshahani らは C/C++で記述した CNN の Vitis を用いた性能を予測するため，400 パターンのパラメータの組み合わせで試して学習させた [7].Calore らは Vitis を使った際の様々なアプリケーションの性能予測をするために，ルーフラインモデル用にピーク性能とメモリバンド幅を変えられる計算カーネルを作った [8].

本研究では，1 つのアプリケーションについて SDSoC と Vitis のツール間の比較をする点，また最大性能を出すパラメータを推定する点がこれらの研究と異なっている．

3 重力多体問題

本研究では FPGA による数値アクセラレータの対象として重力多体問題を扱う．重力多体問題は何もない宇宙空間に質点をばら撒いたとき，その後の運動が万有引力によってどう変化するものかを問うものである．三体問題以上は解析的な算出が困難であることから，差分法による計算技術を用いる．重力多体問題計算では質点に働く力の計算が最も大変であるため，本研究では力の計算を行う部分のみハードウェア化する．

質点 j が質点 i に及ぼす万有引力の x 成分を $f_{x_{ij}} \cdot m_i$ とすれば，

$$f_{x_{ij}} m_i = G m_i m_j \frac{x_i - x_j}{r_{ij}^3}. \tag{1}$$

ここで，r_{ij} は質点 i と質点 j の距離，m_i, m_jはそれぞれ質点 i と質点 j の質量，G は万有引力定数である．これより，単位質量あたりの質点 i が受ける万有引力の x 成分の合計 f_{x_i} は，

$$f_{x_i} = \sum_{j=0}^{N} f_{x_{ij}}. \tag{2}$$

ここで N は質点の数である．これを y 方向と z 方向においても同様にして求めることで重力多体問題を計算することが出来る．

本論文では計算速度として Gflops の単位を使っているが， 1 ペアの粒子間の力の計算に 38 演算を要していると仮定して flops に換算している．また，平方根や逆数の演算があってそれらは加算や乗算より多くの時間がかかることから重力多体問題分野での慣例として 38 の数字を使用している．

4 SDSoC と Vitis の比較

4.1 パイプラインの計算性能

FPGA での専用ハードウェアによる 1 回の処理時間を T_{fpga} とした時，

$$T_{fpga} = t_{fpga} \times N^2 \tag{3}$$

と表すことが出来る．ここで t_{fpga} は 1 ペアの粒子の力の計算にかかる時間である (表 1).

4.2 通信性能

専用パイプラインはアプリケーションプログラムからデータを受け取り処理結果を返す必要がある．1 回の計算ルーチンの呼び出しに対応する通信時間を T_{comm} とすると，

$$T_{comm} = t_{band} \times 32 \times N + 2 + t_{lat} \tag{4}$$

と表すことが出来る．ここで t_{band} は 1byte を転送するのにかかる時間，t_{lat} は一回の転送を始める際のセットアップ時間である．1 粒子の座標は 16byte, 1 粒子に働く力も 16byte のデータが必要であり，座標と力の 2 回の転送を必要とするため式 (4) のようになる．

転送速度を測るルーチンを作成して計測したところ，t_{band} , t_{lat} は表 1 のようになった．

4.3 全体性能

サブルーチンを 1 回処理する時間 T は，

$$T = T_{fpga} + T_{cpu} + T_{comm} \tag{5}$$

と表せる．ここで T_{cpu} は

$$T_{cpu} = t_{cpu} \times N \tag{6}$$

と表せる．t_{cpu} は表 1 のようになった．式 (3)-(6) から計算速度を求めたものが図 1 である．曲線は T を，点は実測値を意味する．

表 1: 処理時間の実測値

	SDSoC	Vitis	Vitis(OpenCL)
t_{fpga}	3.4×10^{-8}	6.7×10^{-8}	5.7×10^{-8}
t_{cpu}	2.7×10^{-7}	2.0×10^{-7}	3.0×10^{-7}
t_{band}	2.6×10^{-9}	7.9×10^{-11}	1.5×10^{-9}
t_{lat}	4.8×10^{-4}	0	6.3×10^{-4}

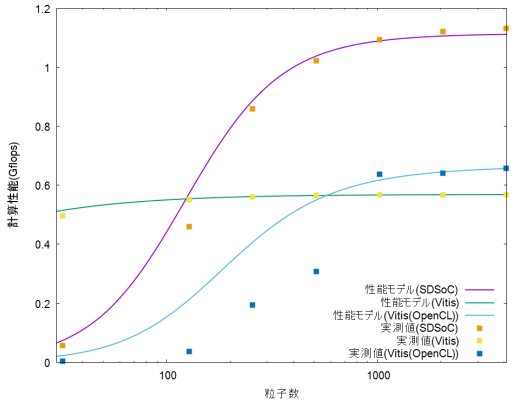

図 1: 実測方法の違いによる計算性能

4.4 手動でのループアンローリング

プラグマを用いたループアンローリングを行ったが計算性能の向上には至らなかったので手動でループアンローリングを行った．図 2 に計算性能の比較を示した．この結果より，計算性能はループ展開の数におよそ比例して向上することが分かった．しかし，回路資源にはまだ余裕があるがループ展開を 64 にするとコンパイルに失敗してしまった．

4.5 複数パイプライン

前節のように SDSoC も Vitis も HLS の機能でループアンローリングを行って並列処理をすることが可能であるが，ここでは（データ転送を含んで）関数全体を複数並列化したい場合を考える．SDSoC では関数呼び出しの部分に特有の SDS プラグマを使用することで非同期で計算ユニットを呼び出し並列に計算させることが容易であった．

一方，Vitis では SDS プラグマは使用できないので OpenCL を用いてアウトオブオーダー実行を指定することで並列処理を行うことが出来る．コードを大きく書き換える必要があるため，プラグマのように普通のコンパイラでコンパイルするだけで元のコードが走るような便利さは無くなる．図 3 に計算性能の比較を示した．この結果より，並列数におよそ比例して計算性能は向上することが示された．

5 最適パラメータ推定

前章までの調査をもとに最適パラメータを推定する．本研究では最適なパラメータとしてループ展開と複数パイプラインの両手法の最適パラメータを推定する．図 4 に粒子数が 256 で LUT が上限の半分しか使うことが出来ない場合の推定値を示した．この図より粒子数が少ない場合では複数パイプラインを用いた並列化の方が優位なのでループ展開を 3，パイプラインを 9 分割する時が，最大性能であると推定できた．

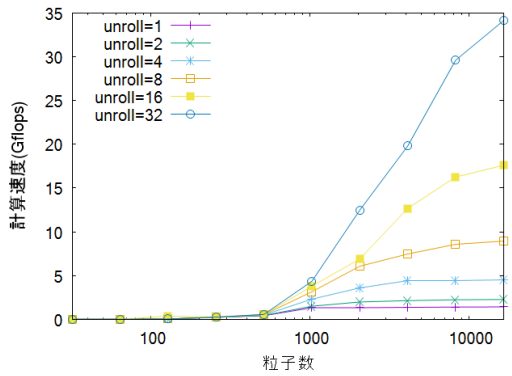

図 2: 手動でのループアンローリングによる計算性能比較

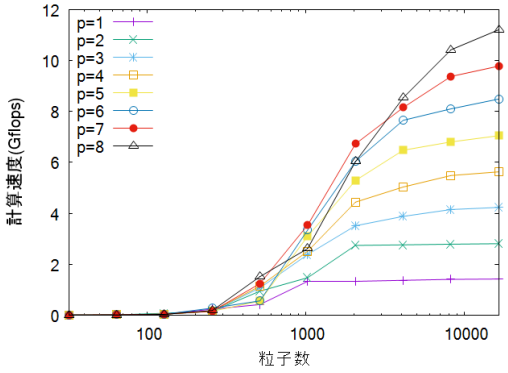

図 3: 複数パイプラインの計算性能比較

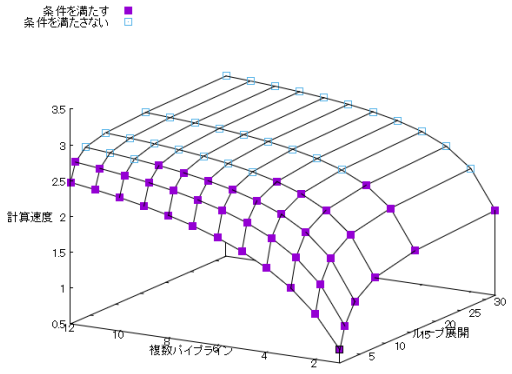

図 4: 複数パイプラインとループ展開を用いた際の最大性能予測

6 おわりに

本稿では，重力多体問題を例として 2 つの高位合成ツールによって実装，比較し評価した．Vitis での実装は，SDSoC とパイプライン性能及び通信性能で違う特徴を示した．しかし，OpenCL を用いて実装をすると SDSoC と似たような挙動を示すことが分かった．

また性能予測について，複数パイプラインでの並列化とループ展開での並列化のどちらが最適となるか性能モデルから推定した．限定的ではあるが複数パイプラインの利点である通信部分も含めた並列化から通信時間の全体に占める割合が大きいときには複数パイプラインでの実装をするとよいという予測が得られた．

今後は予測と実測のずれを評価する必要がある．

参考文献

[1] Xilinx 社，”SDSoC 開発環境”. https://japan.xilinx.com/products/design-tools/software-zone/sdsoc.html (アクセス日 2021-7-10)

[2] Xilinx 社，”Vitis 開発環境”. https://japan.xilinx.com/products/design-tools/vitis.html (アクセス日 2021-7-10)

[3] Akira JINGUJI, Shimpei SATO, and Hiroki NAKAHARA, An FPGA Realization of a Random Forest with k-Means Clustering Using a High-Level Synthesis Design, IEICE Transactions on Information and Systems, E101.D(2), pp. 354-362, 2018

[4] Sam Skalicky, Joshua Monson, Andrew Schmidt, and Matthew French, Hot & Spicy: Improving Productivity with Python and HLS for FPGAs, 2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), pp.85-92,2018

[5] Amin Kalantar, Zachary Zimmerman, and Philip Brisk, FA-LAMP: FPGA-Accelerated Learned Approximate Matrix Profile for Time Series Similarity Prediction, 2021 IEEE 29th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), pp. 40-49, 2021

[6] Jin Wang and Shenshen Gu, FPGA Implementation of Object Detection Accelerator Based on Vitis-AI, 2021 11th International Conference on Information Science and Technology (ICIST), pp. 571-577, 2021

[7] Masoud Shahshahani and Dinesh Bhatia, Resource and Performance Estimation for CNN Models using Machine Learning, 2021 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), pp. 43-48, 2021

[8] Enrico Calore and Sebastiano Fabio Schifano, Performance assessment of FPGAs as HPC accelerators using the FPGA Empirical Roofline, 2021 31st International Conference on Field-Programmable Logic and Applications (FPL), pp. 83-90, 2021