C-003

高位合成を用いて記述したFPGA上の倍精度浮動小数点演算器の論理合成最適化と性能評価

Logic Synthesis Optimization and Performance Evaluation of Double-Precision Floating-Point Arithmetic Units on FPGA Using High-Level Synthesis

周東 裕也† 大本 裕真† 窪田 昌史† 弘中 哲夫†
Yuya SHUTO Yuma OMOTO Atsushi KUBOTA Tetsuo HIRONAKA

1. はじめに

Field-Programmable Gate Array（FPGA）を用いたハードウェアアクセラレータは，高性能かつ消費エネルギー効率の観点から高性能計算の分野で期待されている．特に数値計算の分野では倍精度浮動小数点演算が頻繁に使用されるが，AMD 社の高位合成ツールである Vitis HLS を用いたハードウェア設計においては，浮動小数点演算に時間がかかるという課題が存在する．これに対して，X.Wang らの研究 [1] などで浮動小数点演算器単体での高速化の取り組みがされている．

本研究では，LogiCORE IP[2] が対応する IEEE754 規格に準拠した乗算器および加減算器を HLS C で記述し，記述した演算器をアプリケーションにインライン展開して論理合成を行った．この手法により，高位合成ツールが生成する LogiCORE IP の浮動小数点演算器と比較して浮動小数点演算器が細かな粒度でスケジューリングされ，アプリケーション全体のスケジュールを考慮した最適化が可能であることを確認した．

2. 浮動小数点演算器の実装

LogiCORE IP を設計に用いた結果のタイムチャートを Schedule Viewer で確認すると，浮動小数点演算器が 1 つの単位としてスケジュールされていることが分かる．そのため，浮動小数点演算器を複数の小さな単位に分割して柔軟にパイプラインステージを組む最適化ができていない．

そこで，本研究では LogiCORE IP の仕様に準拠した演算器を C++で設計し，LogiCORE IP を置換できるようにした．この新しい演算器は，浮動小数点演算器を分割可能とすることで最適なパイプラインステージ分割を可能にする．

この演算器は HLS C 設計を用いて作成したアプリケーションに HLS C で記述したソースとして取り込み，アプリケーションから関数呼び出しによって使用できるようにした．また，HLS C 記述した演算器を単に関数として使用すると演算器単位でのスケジューリングが行われるが，関数をインライン展開して使用することで演算器内部の処理単位でのスケジューリングが可能となる．これにより，より細かい粒度でのアプリケーション全体のパイプラインステージを最適化できる．

† 広島市立大学大学院情報科学研究科 Graduate School of Information Sciences, Hiroshima City University

3. 評価

演算器単体の性能、および、10 元連立一次方程式を Gauss-Jordan 消去法で処理したときの処理速度の評価を行う．具体的には，アルゴリズム中の減算および乗算に対して，Vitis HLS の最適化ディレクティブで LogiCore IP の実行レイテンシを指定した場合と，作成した HLS C 記述の演算器を用いた場合で効果を比較する．

さらに，アプリケーション全体のスケジュールを考慮したパイプラインステージの最適化がどのように効果をもたらすかについても評価する．本研究で使用した環境は，表 1 の通りである．

表 1 本研究での合成環境・条件

合成ツール	Vitis HLS 2022.1
対象 FPGA	AMD Zynq UltraScale+ MPSoC ZCU104
クロック周波数	100 MHz
実装手法	C++ 言語による高位合成

3.1 演算器単体の性能評価

評価は，単純に LogiCore IP を用いた場合，LogiCore IP の演算器に実行レイテンシを指定するため BIND_OP プラグマを適用した場合，および HLS C で記述した演算器を使用した場合について行った．それぞれの Vitis HLS による合成結果を表 2 に示す．

はじめに，プラグマを指定しない場合，Fmax が 100MHz を十分に満たす回路が出力された．この演算器は，Vitis HLS のデフォルトで合成される LogiCore IP で実現される演算器であり，評価の基準となる．次に，提案した演算器を使用した場合，Fmax が 100MHz を満たし，実行サイクル数が LogiCore IP に対し、2〜3 サイクル短縮されていることが確認できた．最後に，BIND_OP プラグマを使用して，LogiCore IP 演算器の実行レイテンシを HLS C で記述した演算器の実行サイクル数と同じに設定し，使用リソースに制限をかけずに合成を行った．

これらの結果から，作成した HLS C 記述の浮動小数点演算器は実行速度において優れた性能を示していることが明らかとなった．

表 2 演算器単体での合成結果と評価

演算	latency (cycle)	Fmax (MHz)	impl	DSP	FF	LUT
LogiCORE (BIND_OP プラグマなし)						
加減算	5	197.28	fulldsp	3	450	813
乗算	5	142.01	maxdsp	11	304	234
LogiCORE (BIND_OP プラグマあり)						
加減算	3	78.98	fabric	0	331	830
	3	63.89	dsp	3	448	802
	3	63.89	fulldsp	3	448	802
乗算	2	70.20	fabric	0	479	2,655
	2	63.89	dsp	11	301	217
	2	63.89	meddsp	9	325	270
	2	63.89	fulldsp	10	284	227
	2	63.89	maxdsp	11	301	217
HLS C 記述で作成した演算器						
加算	3	137.58	-	0	347	1,697
減算	3	137.17	-	0	460	1,789
乗算	2	142.63	-	8	123	577

3.2 Gauss-Jordan 消去法の評価

本節では，10 元連立一次方程式に対応する Gauss-Jordan 消去法の実行速度を評価する．アルゴリズム中の減算および乗算に対して，以下の 4 つのアプローチを比較した：①LogiCore IP においてレイテンシを指定する BIND_OP プラグマなし，②あり，③HLS C 記述した演算器をインライン展開せず用いた場合，④インライン展開した場合．それぞれの評価結果を表 3 に示す．

BIND_OP プラグマを使用して，HLS C 記述で作成した演算器と同等のレイテンシを指定した場合（②），演算器がクリティカルパスとなり，Fmax が 100MHz に達しなかった．一方，HLS C 記述で作成した演算器をインライン展開せず使用した場合（③），Fmax が 100MHz を満たす回路が生成された．最適化無しの場合（①）と比較して，乗算では 1.22 倍の高速化が確認され，減算と乗算の両方を適用した場合には 1.42 倍の高速化が見られた．ただし，減算器のみに適用した場合関数呼び出しやパイプライン化のオーバーヘッドにより，高速化は見られなかった．

オーバーヘッドを減らすためにインライン展開を行った結果，最適化無しの場合（①）と比較して，乗算器を適用した場合には 1.43 倍，減算器を適用した場合には 1.18 倍の高速化が達成された．減算および乗算の両方に演算器を適用した場合，1.63 倍の高速化を実現した．また，作成した演算器の適用による資源量の増加は，最適化無しの場合（①）と比較して，演算器に最大限の最適化を行った場合（④）に LUT が約 2.4 倍であった．

3.3 スケジュール最適化の効果

本節では，アプリケーション全体のスケジュールを考慮したパイプラインステージの最適化の効果を評価する．表 3 の結果から，インライン展開を行わない場合（③）とインライン展開を行う場合（④）を比較すると，インライン展開を行った場合（④）に実行速度が向上することが確認できる．特に減算器に注目すると，インライン展開を行わずに HLS C 記述した演算器を適用した場合（③）は最適化無しの場合（①）と同じ実行速度であるが，インライン展開を行う事でレイテンシが 14,138 サイクルから 11,938 サイクルへ約 1.2 倍の高速化が確認された．これらの結果から，HLS C 記述した演算器を用いたパイプラインステージの最適化が，アプリケーション全体の処理速度を向上させることが確認された．

表 3 Gauss-Jordan 消去法の合成結果と評価

最適化	latency (cycle)	Fmax (MHz)	BRAM	DSP	FF	LUT
① LogiCORE (BIND_OP プラグマなし)						
無し	14,138	139.61	4	25	2,577	3,515
② LogiCORE (BIND_OP プラグマあり)						
乗算	10,838	70.20	4	3	2,927	8,364
減算	13,038	78.98	4	22	2,586	3,534
乗算+減算	9,739	70.20	4	0	2,936	8,383
③ HLS C 記述で作成した演算器						
乗算	10,938	139.61	4	11	2,265	3,989
減算	14,138	139.61	4	22	2,850	4,773
乗算+減算	9,938	139.61	4	22	2,469	5,259
④ HLS C 記述で作成した演算器をインライン展開						
乗算	9,838	139.61	4	19	2,470	4,630
減算	11,938	139.61	4	22	3,017	6,926
乗算+減算	8,638	137.02	4	16	2,779	8,526

4. まとめ

本研究では，倍精度浮動小数点演算器を HLS 記述しアプリケーションに適用することで LogiCORE IP を用いる場合に比べ高速な回路が合成できることを確認した．また，演算器をインライン展開することでアプリケーション全体のスケジュール最適化を行うことができ，性能が向上することを確認した．

参考文献

[1] Xiaojun Wang and Miriam Leeser. VFloat: A Variable Precision Fixed- and Floating-Point Library for Reconfigurable Hardware. ACM Trans. Reconfigurable Technol. Syst. 3, 3, Article 16 (September 2010), 34 pages.

[2] Xilinx,“Floating-Point Operator v7.1 Product Guide”, PG060（7.1 English），2020-12-16.

[3] 鈴木昌治,“ディジタル数値演算回路の実用設計,” 谷智美（編），CQ 出版株式会社，東京，2006.

[4] Xilinx,“Vitis 高位合成ユーザー ガイド ”，UG1399（2022.1 Japanese），2022-06-07 .