Vol.2009-SLDM-140 No.4
2009/5/20

自動パイプライン化を用いた FPGA におけるプロトタイピングの高速化

鄭カイ† シンウェイジェイ† 木村智生†† 甲斐康司††
九黒丸俊一†† 木村晋二†

概要 : 本報告では、FPGA におけるプロトタイピングの新しい高速化アルゴリズムを提案する。基本的なアイディアは、FPGA マッピングの結果であるオリジナル回路を分割し、パイプラインレジスタを挿入し、スループットを向上することである。この時、FPGA のロジックエレメントの中の使われていないレジスタを利用し、パイプラインを構成するときに必要な FPGA 資源を削減することを目標とする。分割においては、カットセットベースのアルゴリズムを用いている。本手法をベンチマーク回路に適用し、クロックの 11.9%～109%の向上を確認した。

FPGA-Based Prototyping Acceleration Using Automatic Pipeline Synthesis

Kai ZHENG† Weijie XING† Tomoo KIMURA†† Koji KAI††
Shun-ichi KUROMARU†† Shinji KIMURA†

Abstract: In this report, we propose a new approach for the FPGA-based prototyping acceleration. In this method, a circuit mapped on FPGA is divided into two parts and converted to a pipeline circuit by inserting registers. With this, the throughput of the circuit can be improved. When inserting pipeline registers, we devise a method to use un-used registers in logic elements of FPGA for reducing the resource of the FPGA. A cut-set based algorithm is used in the partitioning. The algorithm is applied to several benchmark circuits, and 11.9% to 109% increases on clock frequency are obtained.

† 早稲田大学大学院 情報生産システム研究科、〒808-0135 北九州市若松区ひびきの 2－7
†† パナソニック（株）、〒814-0001 福岡市早良区百道浜 2－4－16

† Waseda University, Graduate School of Information, Production and Systems, 2-7 Hibikino, Wakamatsu-ku, Kitakyushu-shi, Fukuoka 808-0135, Japan
†† Panasonic Corporation, 2-4-16 Momochihama, Sawara-ku, Fukuoka City 814-0001, Japan

1. はじめに

プロトタイピングは機能検証とハードウェア/ソフトウェア協調設計を短時間で終わらせるための重要な技術である。FPGA におけるプロトタイピングは、プロトタイピング環境の構築が非常的容易であること、再利用性が高いため、幅広く使われている。設計された回路は FPGA にマッピングされ、与えられたパターンに対して実行される。回路の実行時間は FPGA デバイス及びマッピング手法に依存する。近年、FPGA デバイスが高速化しているが、プロトタイピングの速度が十分とは言えない。そこで、パイプライン自動生成を用いた回路の高速化手法について研究を行う。

パイプライン自動生成技術というのは、組合せ回路部分をいくつのステージに分け、その間にパイプラインレジスタを挿入することで、組み合わせ回路の最大遅延を削減する手法である。これにより、クロックの高速化と同時に、スループットを向上でき、全体の処理時間を削減できる。ただし、パイプラインレジスタなど付加的なハードウェア資源が必要となる。FPGA におけるプロトタイピングでは、ロジックエレメントの中の使われていないレジスタをパイプラインレジスタとして利用することで、付加回路を削減できる場合がある。これは、図 1 に示すように、組合せ回路を構成する場合には FPGA のロジックエレメントを構成するルックアップテーブルとレジスタの内、ロジックエレメントのみ用いられ、レジスタは使われないままであることによる。

パイプライン自動生成の既存研究はいくつあるが[(2),(3),(5)~(10)]、ほとんどの研究はパイプラインの高位構造に注目し、パイプラインの対象は単機能の算術演算回路か、CPU や DSP のような命令セットに基づくプロセッサであり、一般的な回路の最適化、特に合成後のゲートレベルの回路に適用することは困難であった[(11)]。

そこで本報告では、ゲートレベルの回路を自動的にパイプライン化する手法を提案する。この手法では、まず回路の論理素子の段数を計算し、段数とデータの依存関係を考慮し、分割およびレジスタを入れる場所を決める。具体的には、段数がほぼ半分になるように回

Vol.2009-SLDM-140 No.4
2009/5/20

路を二分割する。実際は、カットセットと呼ばれるグラフを二つ分ける枝（ワイヤ）の集合を求める。

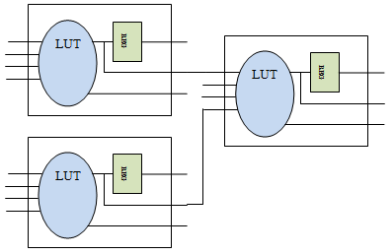


図 1. FPGA のロジックエレメントで表す組合せ回路

2. 定義

節点の集合 V と V×V の部分集合である枝集合 E の二項組みである有向グラフ (V, E) を考える。枝 e = (vi, vj) は節点 vi を出て vj に入ることを表し、e は vi と vj を接続すると呼ぶ。枝で接続された節点と接続枝を並べたもの v1, e1, v2, e2, ..., , en-1, vn (ei = (vi, vi+1)) を歩道（鎖・ウォーク）という。辺の重複を許さない場合、路といい、頂点の重複も許さない場合、道（パス（開いた歩道をパスという場合は単純パス））という。また、始点と終点が同じ路のことを閉路、始点と終点が同じ道（つまり v1, e1, v2, e2, ..., en-1, vn, en, v1 という路で ei が相異なるもの）のことを閉道（ループ）という。

グラフの頂点を節点間に道が存在しない 2 つの部分集合への分割について述べる。 G = (V, E) がグラフを表すとする。E の部分集合 C について、E から C を除くことで、V の節点が接続線を持たない二つの独立な部分集合に分かれる場合、C をカットセットと呼ぶ。一方、節点集合 V を 2 つの共通部分を持たない集合 S と T に分割した場合、S と T (あるいは T と S) を接続するすべての枝の集合はカットセットとなる。カットセットに含まれる枝をカットエッジと呼ぶ。

3. パイプライン自動生成のプロセスについて

自動パイプライン化においては、組合せ回路を有向きグラフと見て、そのカットセットを求めることで、回路を二分割するという手法を用いる。実際には、組合せ回路だけではなく、順序回路を扱う必要がある。順序回路は組合せ回路とレジスタから構成されると考えられるので、与えられた回路の組合せ回路部分、すなわちレジスタとレジスタで挟まれた組合せ回路部分に対してカットセットを求め、そこにパイプラインレジスタを入れることでパイプライン化を行う。カットセットの選定においては、各論理ゲートの段数をベースとし、パイプライン化後の回路の最大段数が元の回路の半分になるようにする。

具体的なプロセスは、以下の三つステップに分けることができる。段数(遅延)の計算、カットセットの選択、レジスタの挿入、の 3 つに分けられる。以下では回路の遅延評価に段数を用いているが、詳細な遅延情報の利用も容易である。なお、レジスタを入れた後の回路の機能は、元の回路と同じでなければならない。これはとくに順序回路中にレジスタを含むループがある場合に問題となる。

3.1 段数(遅延)の計算

カットセットの選定は各論理素子の遅延に基づいているので、ここでは、論理素子および素子間の接続に対して段数を計算し、その結果を使用する。組合せ回路に対し、素子の最大段数は以下の式で求められる。

$\max_d(I) = 0$ if I が外部入力

$\max_d(v) = \max_i \{\max_d(u_i)\}$ u_i は v の i 番目の入力に接続されている素子

すなわち、外部入力の段数は 0、そうでない素子の段数は、入力となる素子の段数の最大値に 1 を加えたものである。

順序回路に対しては、外部入力と同時にレジスタの出力の段数を 0 とすれば、素子の段数を上記の定義に従って計算できる。接続線の段数は、それを駆動する論理素子の段数とする。

3.2 カットセットの選定

レジスタの挿入場所により、パイプライン化の効果は異なる。そこでパイプライン化後に組合せ回路部の最大遅延が約半分になるように二つの部分に分ける以下のアルゴリズムを提案した。

a. 段数計算に基づき、、一番大きい段数の素子の段数を Dmax とする。

b. 段数が Dmax/2 に近い接続線をカットセットとする。具体的には以下の処理を行う。

(ア) 接続線の段数が Dmax / 2 であれば、この接続線をカットセットに入れる。

(イ) 接続線 e = (u, v) について、e の段数が Dmax / 2 より小さくかつ v の段数が Dmax / 2 より大きいならば、e をカットセットに入れる。

(ウ) 接続線 e = (u, v) について、e の段数が Dmax / 2 より小さくかつ v が外部出力あるいはレジスタの入力であれば、e をカットセットに入れる。

これで、カットセットの選定ができるが、実際には一つの素子の出力が複数の素子に接続されるファンアウトに注意をする必要がある。(u, v) と (u, w) のファンアウトについて、一方がカットセットに含まれる場合は、他方についてカットセットに含まれるかどうかチェックを行い、挿入されるレジスタが一つで済むようにする。

3.3 レジスタの挿入

カットセットを選んだ後、一つずつのカットエッジにレジスタを挿入する。ASIC では、レジスタの挿入はにより、回路面積は増えるが、本報告で対象とする FPGA の場合は、レジスタ挿入により回路資源が増えるとは限らない。

FPGA は、図 2 に示すようなロジックエレメントを基本素子としている。各基本素子にはレジスタが配置されているが、ロジックエレメントが組合せ回路の一部として使用されている場合は、このレジスタは使われない。レジスタが使われていない場合は、そのレジスタをオンにするだけで、追加の資源を消費せずにレジスタ挿入ができる。

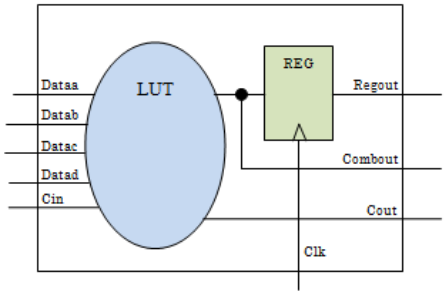


図 2. ロジックエレメントの構造

なお、先に述べたように、ある論理素子の出力から出ている枝(ファンアウト)すべてがカットセットに含まれている場合は、挿入するレジスタは一つで済むが、ファンアウトの一部しかカットセットに含まれない場合は、ロジックエレメントを組合せ出力とレジスタ出力の二種類で使う必要がある。図３にはこの二つの状況を示す。

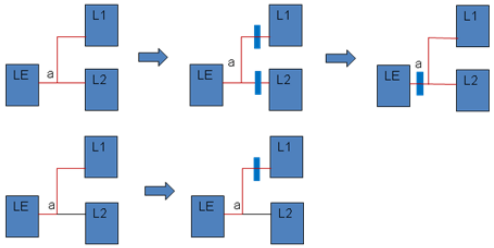


図 ３．レジスタ挿入の資源削減

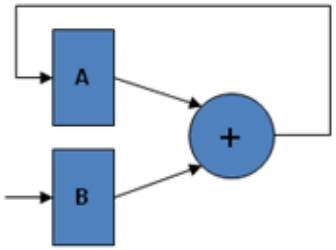
3.4 FPGA デバイスの特徴の活用

最近の FPGA では、ロジックエレメントのパラメータを設定することで、組合せ出力とレジスタ出力を同時に使うことができる。ファンアウト問題に対しては、この特徴を用いて対処を行うことで、パイプライン化時の資源を削減できる。組合せ出力のワイヤがカットエッジに選ばれた時、すべてのファンアウトワイヤが選ばれたら、前のロジックエレメ

ントのレジスタを使って出力する。一部のファンアウトが選ばれたら、選ばれた部分は前のロジックエレメントのレジスタ出力とし、選ばれなかった部分は組合せ出力を使う。

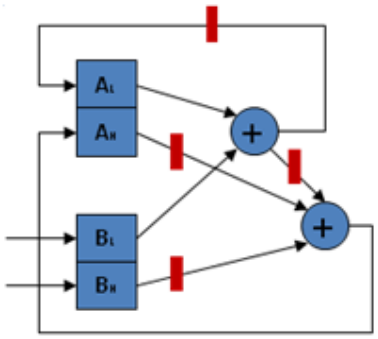
4. ループの存在する回路に対して

単純な組合せ回路およびデータが入力から出力へ流れる一方向の順序回路には、提案アルゴリズムが適用できるが、(順序的な)ループがある場合には、パイプライン化を行っても高速にならない場合がある。例えば、図４のような F=x1+x2+x3+…を実現する回路を考えると、レジスタ A に前回の結果を保存し、B に取り込んだ入力との加算を行う。提案アルゴリズムを適用すると、図５のように組合せ部分にレジスタが挿入されるが、これはパイプラインとして動作させることはできない。なぜならば、A に正しい加算結果が入るまでに 2 クロック必要であるためである。このため 1 クロックずつ入力データを供給するのでは結果が正しくなくなる。このように、ループがある場合はパイプライン化により動作がおかしくなる。これは、データを正しいタイミングで供給する問題で、データフォワーディングの自動化問題と捉えることができる。具体的には、図６に示すように加算器から AL への接続上のレジスタと AH から加算器への接続上のレジスタを削除すれば、パイプライン的に動作する。ただし、最終的な積算の結果を取り出すには AL の結果を 1 クロック遅らせて出すなどの付加回路が必要である。



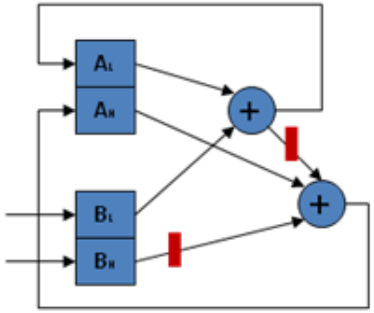
Clock	0	1	2	3	4
A	0	1	3	6	10
B	1	2	3	4	5

図 4. F＝x1+x2+x3+...を実現する元回路



Clock	0	1	2	3	4
AL	0	0	1	2	4
AH	0	0	1	2	4
BL	1	2	3	4	5
BH	1	2	3	4	5

図 5. アルゴリズム修正前の結果



Clock	0	1	2	3	4
AL	0	1	3	6	10
AH	0	0	1	3	6
BL	1	2	3	4	5
BH	1	2	3	4	5

図 6. アルゴリズム修正後の結果

このヒューリスティックなデータフォワーディングに基づく最適化手法は、順序的なループの中に含まれる枝をカットしないという方式と一般化できる。ループ内の枝をカットセットに含めないという手法は、パイプラインが正しく動作するための十分条件となっている。本手法を実現する上では、段数を計算する前に回路の中のループを識別してループ内の接続枝をマークし[1],[12]、カットセットを作る時にマークされた接続枝をカットエッジに選ばないという方式を用いている。この方式ではループ部分の遅延(段数)がパイプライン化による高速化の効果を限定する。

5. 実験と結果

提案このアルゴリズムの有効性を証明するには示すため、アルゴリズムを Microsoft Visual Studia 2008 の C# を用いて実現し、いくつかの複数の回路に適用して実験を行ったは行われた。実験では Altera 社の FPGA 用の設計環境（Qariutus）で用いられている VQM（Verilog Quartus Mapping）ネットリストファイルを元の電回路として用い、パイプライン化を行った後の回路を VQM として出力して、再び Altera 社 Quartus II 8.0 を用いてマッピングし、FPGA にマッピング後のリソース数とクロック周波数を評価している。FPGA としては Stratix I アーキテクチャに基づくものを用いた。

以下は VQM での一つのロジックエレメントの記述例である。入力、出力とロジックエレメントの使用状況を記述する。lut_mask の 9966 は Look-up table の真理値表を表す。

```
Stratixl_cell ¥fulladder:fa0|Add1~66_I (
.dataa(b[0]) ,
.datab(a[0]) ,
.datad(cin) ,
.combout(¥fulladder:fa0|Add1~66 )) ;
defparam ¥fulladder:fa0|Add1~66_I .operation_mode = "Normal";
defparam ¥fulladder:fa0|Add1~66_I .synch_mode = "off";
defparam ¥fulladder:fa0|Add1~66_I .register_cascade_mode = "off";
defparam ¥fulladder:fa0|Add1~66_I .sum_lutc_input = "datac";
defparam ¥fulladder:fa0|Add1~66_I .lut_mask = "9966";
defparam ¥fulladder:fa0|Add1~66_I .output_mode = "comb_only";
```

この素子は組合せ素子であるが、それは最後の行の output_mode というパラメータで指定されている。output_mode としては、reg_only、comb_only、reg_and_comb があり、これによりレジスタ出力のみ、組合せ素子、レジスタと組合せ出力の両方の指定ができる。パラメータ.combout と.regout はその場合の出力名を指定するためのものである。

Microsoft Visual Studio 2008 の C#言語では、VQM の入出力を含めて自動パイプライン化アルゴリズムを実現した。プログラムには VQM ネットリストファイルの入力導入、パイプライン化処理、と導出 VQM の出力という三つの機能を実現した。

加算器と ALU(Arithmetic Logic Unit) に対する実験の結果は、表 1 のようになった。クロックの周波数の改善は平均で 17.4%であった。これは段数を半分にしていることを考えると良い結果ではないが、QuartusII では FPGA 内部のキャリーチェーンを用いて加算を実現することを考えると悪くない結果といえる。

つぎに乗算器について評価を行った。表 2 には、Full Adder を人手で接続し手作成した配列乗算器に対する実験結果を表している。キャリーチェーンが使われていないので、8 ビット、16 ビット、32 ビットすべてで、期待通りクロック周波数が倍になった。

さらに、ISCAS85 の組合せ回路に対しても、実験を行た。結果を表 3 に示す。平均で 55.2%のクロック周波数の増加が得られた。表 4 には ISCAS85 の順序回路の実験結果を示す。順序回路にはループがあり、カットできない部分があるため、周波数の増加は組合せ回路に比較して著しくはないが、平均で 27.5%のクロック周波数の増加が得られた。

6. まとめ

本報告では、自動パイプライン化を用いたＦＰＧＡにおけるプロトタイピングの高速化手法を提案した。本アルゴリズムは、段数(遅延)の計算、カットセットの選択とレジスタの挿入に基づいている。また、回路中に含まれる（順序的な）ループの問題についても、ひとつの解決法を示した。さらに、本アルゴリズムを実現し、Altera Stratix I FPGA の VQM ファイルを入力としてパイプライン化し、新たな VQM ファイルとして生成するシステムを実現した。本システムをいくつかのベンチマーク回路に適用し、回路によっては 2 倍(50%のクロック周波数増加)、順序回路ベンチマークに対する平均でも 27% のクロック増加が達成できた。本アルゴリズムは回路を二分割して高速化するものであり、複数回定期用す

ることも可能である。今後は、アルゴリズムの改善と共に、Stratix IV など、新しい FPGA アーキテクチャへの対応を行いたいと考えている。

表１．加算器と ALU の実験結果

Circuit	Original			Pipelined			Increase
	#LE	#REG	Freq(MHz)	#LE	#REG	Freq(MHz)	
add16	64	48	314.96	83	73	377.5	19.90%
add32	128	96	300.3	163	145	342.94	14.20%
add48	192	144	281.85	243	217	321.75	14.20%
add64	256	64	252.21	323	289	304.41	20.10%
alu8	156	29	149.39	237	126	167.79	12.30%

表２．配列乗算器の実験結果

Circuit	Freq(MHz)	Total LE	Comb	Reg_only	Comb_Reg	Circuit	Freq(MHz)	Total LE	Comb	Reg_only	Comb_Reg
mult8	57.5	187	155	32	0	mult8_p	106.2	237	118	82	37
mult16	23.8	755	691	64	0	mult16_p	43.1	977	554	286	137
mult32	10.2	3043	2915	128	0	mult32_p	21.4	3993	2386	1078	529

表３．ISCAS85 組み合わせ回路の実験結果

Circuit	**Freq(MHz)**	Total LE	Comb	Reg_only	Comb_Reg	Circuit	**Freq(MHz)**	Total LE	Comb	Reg_only	Comb_Reg
c8	**263.4**	82	36	32	14	c8_p	**338.29**	90	14	49	27
C432	**60.1**	109	66	39	4	C432_p	**144.1**	141	48	71	22
C499	**150.3**	149	76	73	0	C499_p	**232.8**	181	68	105	8
C880	**110.8**	199	113	78	8	C880_p	**177.7**	253	71	132	50
C1355	**154.2**	147	74	73	0	C1355_p	**247.1**	179	66	105	8
C2670	**123.8**	335	131	179	25	C2670_p	**225.1**	322	77	179	66
C3540	**68.8**	397	325	54	18	C3540_p	**81.3**	428	302	85	41
C5315	**101.8**	687	402	243	42	C5315_p	**126.6**	718	266	362	90
C6288	**22.1**	591	527	63	1	C6288_p	**38.3**	624	517	74	33
C7552	**101.4**	719	419	263	37	C7552_p	**112.8**	771	349	329	93

表４．ISCAS85 順序回路の実験結果

Circuit	**Freq(MHz)**	Total LE	Comb	Reg_only	Comb_Reg	Circuit	**Freq(MHz)**	Total LE	Comb	Reg_only	Comb_Reg
S298	**263.4**	82	36	32	14	c8_p	**338.29**	90	14	49	27
S1196	**60.1**	109	66	39	4	C432_p	**144.1**	141	48	71	22
S1238	**150.3**	149	76	73	0	C499_p	**232.8**	181	68	105	8
S1423	**110.8**	199	113	78	8	C880_p	**177.7**	253	71	132	50
S1488	**154.2**	147	74	73	0	C1355_p	**247.1**	179	66	105	8

参考文献

1) K.KIM, Y.KANG “Recognition of strongly connected components by the Location of Nonzero elements occurring in C(G) = (D − A(G))−1,” Bulletin of the Korean Mathematical Society, Vol. 41, pp. 125-135, 2004.
2) Daniel Kroening, Wolfgang J. Paul “Automated Pipeline Design” Proceedings of the 38th Conference on Design Automation, pp.810-815, 2001.
3) Alex Panato, Sandro Silva, Flvio Wagner, Marcelo Johann, Ricardo Reis, Sergio Bampi “Design of Very Deep Pipelined Multipliers for FPGAs” Proceedings of the Conference on Design, Automation and Test in Europe, Vol.3, 2004.
4) Sumit Gupta, Nikil Dutt, Rajesh Gupta, Alexandru Nicolau “Loop Shifting and Compaction for the High-Level Synthesis of Designs with Complex Control Flow” Proceedings of the Conference on Design, Automation and Test in Europe, Vol.1, 2004.
5) Nagendran Rangan, Karam S. Chatha “A technique for throughput and register optimization during resource constrained pipelined scheduling” Proceedings of the 18th International Conference on VLSI Design held jointly with 4th International Conference on Embedded Systems Design, pp.564-569, 2005.
6) Maria-Cristina V.Marinescu and Martin.Rinard, “High-level Automatic Pipelining for Sequential Circuits,” In proceeding of System Synthesis, the 14th International Symposium, pp.655–661, 2001.
7) Suryanarayana B. Tatapudi, “ A Mesochronous Pipelining Scheme for High-Performance Digital Systems,” IEEE Transactions on Circuits and Systems, Vol.53, No.5. pp.1078 - 1088, May 2006.
8) Ali Shatnawi, Jehad Ghanim, M.O.Ahmad, “High-level synthesis of integrated heterogeneous pipelined processing elements for DSP applications,” In Proceeding of the Computers and Electrical Engineering, Vol.30, No.8. pp.534–562, Nov, 2004.
9) Tin-Chak-Johnson Pang, Chiu-Sing Choy, Cheong-Fat Chan and Wai-kuen Cham, “Self-timed Booth’s Multiplier,” In Proceeding of ASIC. 2nd International Conference, 1996, pp.280–283, 1996.
10) Tatapudi. S.B and Delgado-Frias, “Designing Pipelined Systems with A Clock Period Approaching Pipeline Register Delay,” In Proceeding of Circuits and Systems, 48th Midwest Symposium, pp.871–874,2005.
11) J. L. Hennessy and D. A. Patterson, “Computer Architecture: A Quantitative Approach,” Morgan Kaufmann Publishing Co., 1st Edition,1990.
12) Alfred V.Aho, John E.Hopcroft, Jeffrey D.Ullmam “Data structures and Algorithms,” Addison Wesley., January 11, 1983.