Vol.2021-ARC-243 No.25
Vol.2021-SLDM-193 No.25
2021/1/26

FPGA向き自己同期型パイプライン回路構成法

吉川 千里[1,a)]　三宮 秀次[1,b)]　岩田 誠[2,c)]　佐藤 聡[1,d)]　西川 博昭[1,e)]

概要：自己同期型パイプラインは，データ処理中のパイプライン段のみが駆動される省電力動作を自然に実現でき，高い性能対消費電力効率が求められるシステムの実現に有望な回路アーキテクチャである．システム開発の要となるプロトタイピングには，柔軟な試作を可能とし，最終生産物にもなり得る商用 FPGA の活用が望ましい．しかし，FPGA とその CAD ツールは同期回路の実装を指向しているため，非同期回路の一種である自己同期型パイプラインは，標準的な回路構成と設計手順では，FPGA 上に実装できなかった．本稿は，Xilinx 社と Intel 社のそれぞれの FPGA を対象に，FPGA の構成要素である LUT を活用した小規模な自己同期型転送制御回路と，それに基づく自己同期型パイプラインの FPGA 実装を可能とする設計手順からなる回路構成法を提案する．実装結果に基づき，提案回路は従来に比べて 50%の回路規模で実現でき，これにより，パイプラインのスループットを最大で約 3.2 倍に向上できることに加え，応用例の一つであるプロセッサのスループットを最大で約 1.6 倍に向上できることを確認した．

1. はじめに

自己同期型パイプラインは，省電力化と高スループット化の両立が求められる IoT(Internet-of-Things) システム等の実現に有望な回路アーキテクチャの一つである．本パイプラインでは，有効データを処理するパイプライン段回路のみが駆動されるため，要求されたスループットを実現するデータ処理にのみ，電力消費が自然に局限される [1]．システム開発の要となるプロトタイピングには，短時間かつ低コストでの回路の試作・評価を可能とするのみでなく，最終生産物にもなり得る，FPGA（Field Programmable Gate Array）の活用が望ましい．しかし，FPGA とその CAD ツールは同期回路の実装を指向しているため，非同期回路の一種である自己同期型パイプラインは標準的な回路構成や設計手順では FPGA 上に実装できなかった．

本研究は，広く利活用されている商用 FPGA である Xilinx 社と Intel 社のそれぞれの FPGA を対象とした自己同期型パイプライン実装法の確立を目指す．既に，回路検証に必要なタイミング制約を定式化するとともに，動作検証に利用できる実遅延付きシミュレーションが一部のデバイスで提供されない Intel 社製 FPGA 向けの設計手順を提案した [2]．本稿では，これを発展させ，自己同期型パイプラインを両社の FPGA 上に実装可能とするとともに，高スループット化を実現可能とする回路構成法を明らかにし，その有効性を示す．

1 筑波大学
University of Tsukuba
2 高知工科大学
Kochi University of Technology
a) s2020661@s.tsukuba.ac.jp
b) san@cs.tsukuba.ac.jp
c) iwata.makoto@kochi-tech.ac.jp
d) akira@cc.tsukuba.ac.jp
e) nishikawa.hiroaki.fp@u.tsukuba.ac.jp

2. 自己同期型パイプライン

2.1 基本構成

自己同期型パイプラインは，図 1 に示すように，データラッチ（DL），機能ロジック（FL），自己同期型転送制御回路（C）からなるパイプライン段を構成単位とする．有効データは，隣接するパイプライン段間で C による，転送要求（send）信号と転送許可（ack）信号を用いた 4 相式ハンドシェイク [3] と呼ばれる一種のネゴシエーションによって転送される．この局所的なデータ転送により，有効データのみが転送される．すなわち，パイプライン段水準のシグナル・ゲーティングが自然に実現され，スイッチングにより電力を消費する回路が真にデータ処理中のパイプライン段に局限され，システム水準の省電力化につながる [4]．4 相式ハンドシェイクは以下の手順により実現される．

- (1)C_{i+1} は $send_i$ 入力信号がアサートされると ack_{i+1} 出力信号をアサートする．
- (2)C_{i+1} の ack_{i+1} 出力信号がアサートされると，前段の C_i は $send_i$ 出力信号をネゲートする．
- (3)C_{i+1} の $send_i$ 入力信号がネゲートされると，C_{i+1} は ack_{i+1} 出力信号をネゲートすると同時に cp_{i+1} 信号をアサートする．これにより，前段から出力された

ⓒ 2021 Information Processing Society of Japan

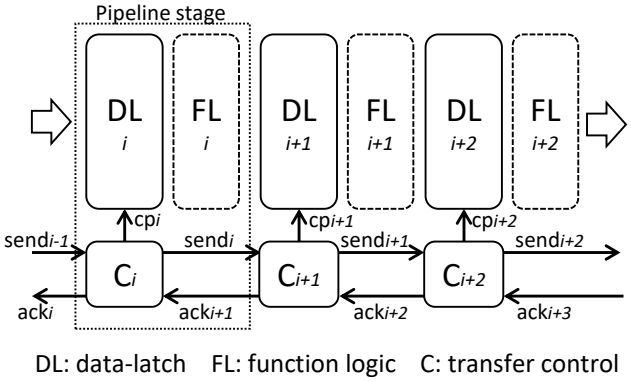

図 1 自己同期型パイプラインの基本構造

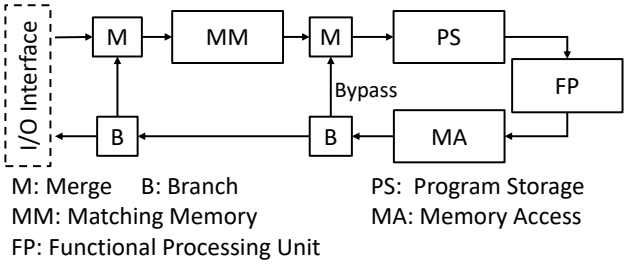

図 2 ULP-CUE のブロック図

データが C_{i+1} が属するパイプライン段のデータラッチにラッチされる．同時に C_{i+1} は $send_{i+1}$ 出力信号をアサートする．(cp_{i+1} 信号は，ack_{i+2} 入力信号がアサートされたときにネゲートする．)

- 同様に後段の C は手順 (1)〜(3) を繰り返す．

2.2 自己同期型データ駆動プロセッサ

自己同期型パイプラインの応用例の一つである自己同期型データ駆動プロセッサは，計算資源の許す限り，演算の実行を対象データが到着した時点で開始するデータ駆動処理方式に基づく [1]．本プロセッサでは，環状の自己同期型パイプラインにより，演算実行に必要な情報を付帯させたデータが互いに独立に処理される．具体的には，演算のデータ対を検出するマッチングメモリ部（MM），演算のフェッチを実現するプログラム記憶部（PS），演算を実行する機能処理部（FP），メモリアクセス部（MA），パイプラインの合流を実現するマージ部（M），パイプラインの分流を実現するブランチ部（B）により構成される．

図 2 に，超低消費電力化データ駆動ネットワーキングプロセッサ ULP-CUE [5] の構成を示す．ULP-CUE は，データ対の検出を必要としない単項演算を MM を迂回して実行できる経路（図 2 中の Bypass）を設けることにより，消費電力と処理時間の削減を実現している．

2.3 自己同期型転送制御回路

自己同期型転送制御回路 C は，図 3 に示すように，非同期回路においてデータ転送制御に用いられるマラーの C 素子 [3] と呼ばれる回路を，標準的な論理ゲートのみを用いてハンドシェイクが実現できるように拡張し，データ転送タイミングを調整する遅延モジュール (Dx) を付帯させた構成をとる [1]．

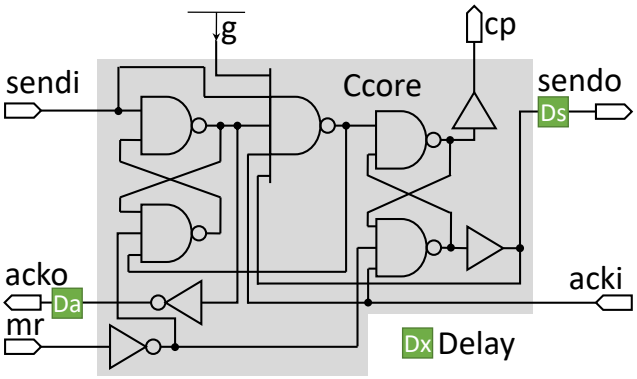

図 3 自己同期型転送制御回路（C）

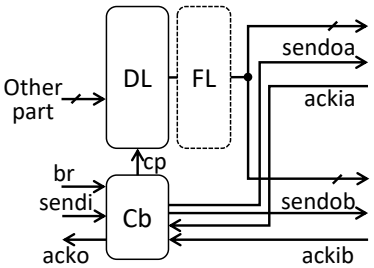

図 4 パイプラインの分流

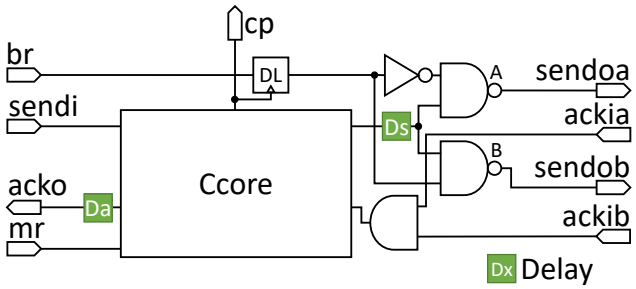

図 5 分流用 C（Cb）

自己同期型データ駆動プロセッサを構成する環状パイプラインには，データの転送に加えて，データの消去及び複製，パイプラインの分流及び合流が必要となる．これらの機能を実現するため，各機能毎に，C の Dx を除く部分（Ccore）を拡張して自己同期型転送制御回路を実現している．具体例として，図 4 に，分流を実現するパイプライン段を示す．分流は，分流方向を示す信号 (br) に従い，後続する二つのパイプライン段のうち一方のみと排他的にハンドシェイクを実施することで実現される．この分流用の自己同期型転送制御回路（Cb）は，図 5 に示すように，Ccore を，br に従いハンドシェイクの方向を制御できるように拡張して実現される．

ⓒ 2021 Information Processing Society of Japan

Vol.2021-ARC-243 No.25
Vol.2021-SLDM-193 No.25
2021/1/26

2.4 関連研究

FPGA 上に非同期回路を実現する研究は複数報告されている．Tranchero らは，回路記述に論理合成及び最適化を制限可能な一種のディレクティブを付加することで，FPGA 上にマラーの C 素子を実装可能とし，非同期回路を実現している [6]．しかし，配置配線及びタイミング検証手法，ならびに，マラーの C 素子以外の回路構成法が示されていない．

Furushima らは，Intel 社製 FPGA を対象として，非同期回路向けの設計制約生成，タイミング検証，遅延調整の自動化を行う設計支援ツールを提案し，これにより自己同期型パイプラインと類似のハンドシェイクを実現する束データ方式プロセッサを FPGA 上に構築する手法を提案している [7]．しかし，この手法では，データ転送に自己同期型転送制御回路 (C) とは異なる制御回路が想定されているため，提案された手法を自己同期型パイプラインに適用できない．

本稿では，Intel 社に加えて Xilinx 社の FPGA も対象として，高スループット化を実現する小規模な自己同期型転送制御回路と，それに基づく自己同期型パイプラインの設計手順を提案する．

2.5 FPGA 実装のための要件

通常の同期回路の FPGA 実装においては，記述された回路の論理合成及び配置配線を実行し，タイミングを検証する手順が一般的である．しかし，この手順により自己同期型パイプラインの実装を試みた場合，以下に示す困難が生じる．

- 一部の回路が縮退され，4 相式ハンドシェイクが実現できない回路が生成され得る．これに対して，FPGA 実装に適した回路構成が必要である．
- CAD ツールが自己同期型転送制御回路のクリティカルパスを検出できない．これに対して，クリティカルパスとそのタイミング制約を明らかにする必要がある．
- 自己同期型パイプラインが持つパイプライン段単位の高いモジュール性を無視した配置配線により，処理遅延が著しく増大し得る．これに対して，自己同期型転送制御回路の配置領域を指定する必要がある．
- 大域的配線の非効率的な割当てにより，処理遅延が増大し得る．これに対して，大域的配線の使用範囲を制御する必要がある．
- CAD ツールによるタイミング検証がサポートされない．これに対して，タイミング制約を独自に検証する必要がある．

3. FPGA 向き回路構成法

3.1 FPGA 向き自己同期型転送制御回路

従来の自己同期型転送制御回路では，図 3 に示すように，NAND ゲート・ラッチを用いて，cp 信号は正論理で，send 信号と ack 信号は負論理で実現されている．NAND ゲート・ラッチは，論理合成時に回路が縮退され，意図した動作が実現されない場合がある．これに対して，send 信号と ack 信号を正論理で実現することにより，異なる論理の出力を実現する素子を設ける必要をなくすとともに，回路の縮退回避と高スループット化につながる小規模化を同時に実現する．具体的には，FPGA 上の組み合わせ回路の構成単位であり，入力に対して任意の論理を実現可能な LUT(Look Up Table）を活用して NAND ゲート・ラッチと同様の機能を有する疑似 SR-FF（QSR）を実現し，これを用いた FPGA 向け自己同期型転送制御回路 C′（図 6）を提案する．QSR は，図 7 に示すように，1 つの LUT（LUTqsr）とフィードバックにより，いわゆる SR-FF（Set Reset Flip Flop）の論理を実現する．LUTqsr の真理値表を表 1 に示す．

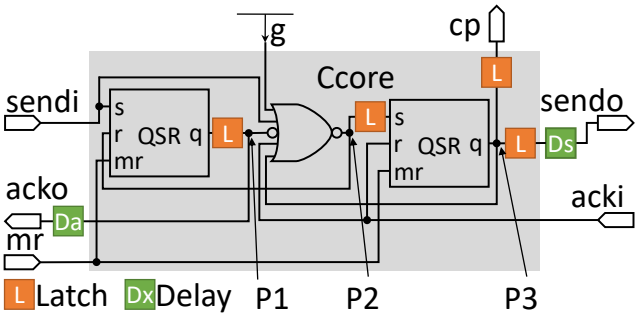

図 6 FPGA 向け自己同期型転送制御回路（C′）

QSR では，入力の変化により出力が変化し，その変化が入力へフィードバックされる前に，再度入力信号が変化すると，発振が起こり得る．しかし，ハンドシェイク中の信号遷移においては，そのような入力の変化が生じないため，提案回路において発振は起こらない．

図 6 中の Dx には，ハンドシェイクのタイミングを調整する遅延モジュールを配置する．具体的には，*LCELL* プリミティブ（Intel FPGA）または *LUT1* プリミティブ（Xilinx FPGA）を直列に接続することで，その数により遅延量を調整する．

3.2 クリティカルパスとタイミング制約

自己同期型パイプラインにおいても同期回路と同様に，データラッチに到着したデータは，データラッチのセットアップ時間が経過した後にラッチされなければならない．パイプライン段 i の DL と FL のクリティカルパス上の信号伝搬時間と，パイプライン段 $i+1$ の DL のセットアップ時間の経過を保証するための時間を T_{fi} と呼ぶ．また，DL がデータをラッチした後，ホールド時間が経過するまでデータを変化させてはならない．パイプライン段 $i+1$ の DL のホールド時間を保証するための時間を T_{ri} と呼ぶ．

T_{fi} と T_{ri} の和 $(T_{fi}+T_{ri})$ は，パイプラインの性能特性

ⓒ 2021 Information Processing Society of Japan

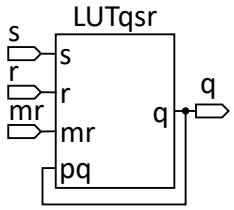

図 7　疑似 SR-FF (QSR)

表 1　LUTqsr の真理値表

s	r	mr	pq	q
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

を決定するとともに，パイプライン段間のデータ転送に要する最小時間，すなわちパイプラインタクトを意味している [8]．T_{fi} と T_{ri} を決定するクリティカルパス上の信号伝搬時間を，図 6 に示した点 P1,P2,P3 を起点として明らかにする．後段とのハンドシェイクは，5 入力 NOR の出力 (P2) の立ち上がりを起点とする．すなわち，P2 の立ち上がりは，後段とのハンドシェイクが開始されたことを意味する．隣接する C′ においてハンドシェイクを実現する信号遷移と，それに要する時間は以下の通りである．なお，マスターリセット信号 mr のアサートにより sendo と acko をネゲートした後，mr をネゲートしてあるとする．

- C'_{i-1} の P2 が立ち上がると，P3 が立ち上がる．これに要する時間を T_{1i} とする．また，この時 cp も立ち上がる．
- C'_{i-1} の P3 が立ち上がると，sendo が立ち上がり，C'_i の sendi が立ち上がり，P1 が立ち上がる．これに要する時間を T_{2i} とする．
- C'_i の P1 が立ち上がると，acko も立ち上がり，C'_{i-1} の acki が立ち上がり，C'_{i-1} の P3 が立ち下がる．これに要する時間を T_{3i} とする．
- C'_{i-1} の P3 が立ち下がると，sendo が立ち下がり，C'_i の sendi が立ち下がり，P2 が立ち上がる．これに要する時間を T_{4i} とする．また，この時 cp も立ち下がる．
- C'_{i-1} の P2 が立ち上がると，P1 が立ち下がる．これに要する時間を T_{5i} とする．また，この時後段とのハンドシェイクが開始される．
- C'_i の P1 が立ち下がると，acko も立ち下がり，C'_{i-1} の acki が立ち下がり， P2 も立ち下がる．これに要する時間を T_{6i} とする．

以上より，T_{1i}〜T_{6i} を構成する経路がハンドシェイクのクリティカルパスである．ここで，C'_{i-1} においてハンドシェイクが開始されてから，C'_i においてハンドシェイクが開始されるまでの時間 $T_{1i}+T_{2i}+T_{3i}+T_{4i}$ が T_{fi} であり，C'_i においてハンドシェイクが開始されてから，C'_{i-1} においてハンドシェイクが開始可能となるまでの時間 $T_{5i}+T_{6i}$ が T_{ri} である．

T_{fi} によりパイプライン段 i の DL と FL のクリティカルパス上の信号伝搬時間と，パイプライン段 $i+1$ の DL のセットアップ時間を保証するための制約は以下の式 (1) で与えられる．

$$T_{fi}+T_{cpi+1} > T_{DLFLi}+T_{Si+1} \tag{1}$$

ここで，T_{cpi+1} は $i+1$ 段目のパイプライン段でハンドシェイクが開始されてから DL に cp_{i+1} が届くまでの時間，T_{DLFLi} は i 段目のパイプライン段でハンドシェイクが開始されてから次のパイプライン段の DL にデータが届くまでの時間，T_{Si+1} は $i+1$ 段目のパイプライン段の DL のセットアップ時間である．

また，T_{ri} によりパイプライン段 $i+1$ の DL のホールド時間を保証するための制約は以下の式 (2) で与えられる．

$$T_{ri}+T_{cpi} > T_{cpi+1}+T_{Hi+1} \tag{2}$$

ここで，T_{Hi+1} は $i+1$ 段目のパイプライン段の DL のホールド時間である．

加えて，消去，複製，分流，合流用 C′ には，消去，複製，分流，合流を制御する機能制御信号と，send と ack を伝えるハンドシェイク信号の間に式 (3) に示すタイミング制約が存在する．

$$T_{tc} \geq T_{fc} \tag{3}$$

ここで，T_{tc} はハンドシェイク信号の伝搬に要する時間であり，この式で制約する対象である．また，T_{fc} は機能制御信号の伝搬に要する時間である．例えば Cb（図 5）では，分流方向を示す制御信号 br が，cp 信号によりラッチされ，続く 2 入力 NAND ゲート A に到達した時点以降に，2 入力 NAND ゲート A の他方の入力にハンドシェイク信号が到達しなければならない．この時 T_{fc} はハンドシェイクが開始されてから br 信号が 2 入力 NAND ゲート A に伝搬するまでの時間であり，T_{tc} はハンドシェイクが開始されてから 2 入力 NAND ゲート A にハンドシェイク信号が伝搬するまでの時間である．もう 1 つの 2 入力 NAND ゲート B についても同様の制約が存在し，データ転送を保証するには，これらを満たす必要がある．消去，複製，合流用 C′ においても同様に，機能制御信号とハンドシェイク信号が入力されるゲートに制約が存在する．

3.3　配置領域指定によるデータ転送時間伸長の抑制

FPGA の CAD ツールは，通常，論理合成時に回路記述中の階層をフラット化した上で論理を最適化する．そのた

め，配置対象の各要素が，どのパイプライン段に属しているのかが考慮されない．さらに，CAD ツールは自己同期型転送制御回路のクリティカルパスが T_{1i}〜T_{6i} を構成する経路であることを検出できないため，T_{1i}〜T_{6i} を構成する回路要素が物理的に離れた場所に配置され得る．これを回避するため，モジュールの配置範囲を指定する機能である，LogicLock（Intel FPGA）または Pblock（Xilinx FPGA）を使用して，各パイプライン段の自己同期型転送制御回路が他の回路と干渉せず，かつ可能な限り近くに配置されるよう，各回路の規模に応じて配置領域を指定する．これにより，T_{1i}〜T_{6i} が不必要に伸長することを抑制できる．

3.4 大域的配線の活用によるデータ転送時間の短縮

cp 信号は，パイプライン段内のすべてのデータラッチに入力されるため，cp 信号の配線に大域的配線を使用することで，配線資源の節約と低遅延化が期待できる．一方，消去，複製，分流用 C′ が生成する cp 信号は，各転送制御回路内に存在する，消去，複製，分流機能制御信号用のデータラッチにも接続されているため，これらを含むすべてのデータラッチに大域的配線を使用して配線することは必ずしも低遅延化につながらない．これは，機能制御信号用データラッチは cp 信号の生成源と物理的に近接しており，それに対する cp 信号は大域的配線を経由しない方が早く到達できると期待されるためである．この機能制御信号用データラッチへの cp 信号の伝搬時間は，式 (3) で表されるタイミング制約において，ハンドシェイク信号伝搬時間の下限となり，T_{fi} にも影響する．

これに対して，機能制御信号用データラッチには，大域的配線を指定しない場合に使用される局所的配線により cp 信号を接続する．その他のデータラッチには大域的配線を指定して cp 信号を接続する．例えば，Cb では，br 信号をラッチする DL には局所的配線を使用し，自己同期型転送制御回路外の全ての DL には大域的配線を指定して cp 信号を接続する．これによりデータ転送時間の短縮が期待できる．大域的配線の指定には *global* プリミティブ（Intel FPGA）または *BUFG* プリミティブ（Xilinx FPGA）を使用する．

3.5 タイミング検証手法

CAD ツールによるタイミング検証では，式 (1)，(2)，(3) で示される制約が検証されないため，これらを独自に検証する必要がある．このために，制約対象となるパスのタイミング情報を抽出する必要がある．CAD ツールは，隣接するレジスタ間のタイミング情報を抽出できる．つまり，自己同期型パイプラインにおいても，レジスタとみなされる DL と，FL からなるデータパスのタイミング情報は同期回路と同様に抽出できる．一方，ハンドシェイクを実現する send 信号と ack 信号が伝搬する経路にはレジスタがない．疑似 SR-FF は含まれるが，これは CAD ツールにレジスタとして認識されないため，そのままではハンドシェイクに関するタイミング情報を抽出できない．

そこで，回路記述に論理的に無視でき，レジスタとして認識されるラッチ（L）を挿入し，クリティカルパスを構成するパスをラッチで挟むことにより，タイミング情報を抽出可能とする．ラッチの実装には *latch* プリミティブ（Intel FPGA）または *LDCE* プリミティブ（Xilinx FPGA）を使用する．図 6 中のラッチ（L）は，本手法を適用した結果である．ラッチのクロック信号として，外部からオープン信号を入力し，常時アサートすることで，論理合成時に縮退されることを回避できる．図中では，簡単化のため，このオープン信号は省略している．

図 8 に，自己同期型パイプラインにおけるタイミング検証手順を示す．まず，以下の手順により，CAD ツールを用いてタイミング検証に必要なタイミング情報を抽出する．

(1) パイプライン回路全体を論理合成する．
(2) タイミング検証時にクリティカルパスを判別できるよう，全てのパイプライン段と自己同期型転送制御回路の階層保持を行う．これにより，配置配線後も階層情報及び配線名が保存される．階層保持設定には，*Design partition*（Intel FPGA）または *flatten_hierarchy* オプションを使用する．
(3) 回路全体を論理合成した上で配置配線する．
(4) *report_timing* コマンドを使用し，検証対象である全てのパスのタイミグ情報を取得する．この操作は CAD ツールでは自動的に実行されないため，タイミング情報抽出スクリプトとして与え，自動的に実行させる．

続いて，各パイプライン段がタイミング制約を満たしていることを検証する．この検証を自動的に実行するため，取得したタイミング情報と，式 (1)，(2)，(3) で示されるタイミング制約に基づき，検証，すなわち大小関係を比較してその結果を出力する独自ツールを作成し，タイミング検証の半自動化を実現した．検証の結果，タイミング制約を満たしていないパイプライン段があれば，その自己同期型転送制御回路の遅延モジュール（Dx）の遅延量を調整し，再度上記の (3) と (4) を行う．全ての制約を満足するまでこれを繰り返す．

4. 評価

商用 FPGA である，Xilinx 社製 Zynq-7000 FPGA と Intel 社製 Cyclone IV FPGA を対象として，提案した回路構成法を評価する．CAD ツールは Quartus Prime Standard 18.1 及び Vivado 2019.2 を使用した．

4.1 回路規模およびスループット

提案回路による回路規模の削減，及び，それによるスループットの向上を定量的に評価するため，提案した設計手順

Vol.2021-ARC-243 No.25
Vol.2021-SLDM-193 No.25
2021/1/26

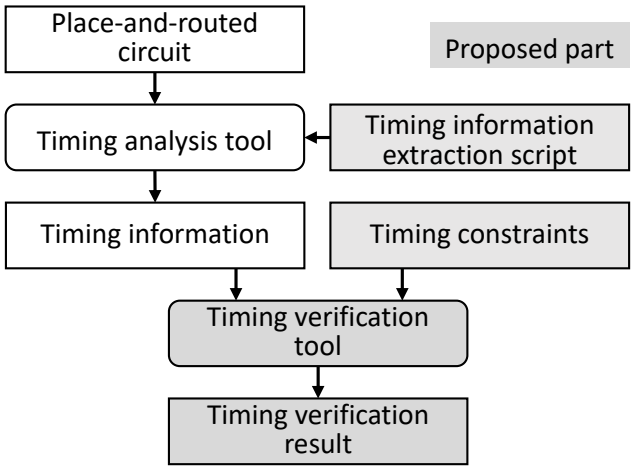

図 8　タイミング検証手順

表 2　自己同期型転送制御回路のリソース使用数

	Cyclone IV	Zynq-7000	
	Logic Cell[個]	LUT[個]	FF[個]
従来回路 C	16	実装不可	実装不可
提案回路 C′	8	3	3
削減率	50%	N/A	N/A

に基づき，従来回路 C と提案回路 C′ を使用した 10 段の直線状自己同期型パイプラインを実装した．スループットを決めるパイプラインタクトは，データパス上の DL と FL に関わるレイテンシに応じて変わる．これらのレイテンシがハンドシェイクにかかる時間内に収まるように理想的にパイプライン分割できた場合，すなわち，最短のパイプラインタクトが実現できた場合のスループットを評価するため，FL と Dx を取り除いた．Zynq-7000 FPGA においては，従来回路 C は実装できなかった．具体的には，CAD ツールによる配置配線が実行できなかった．

パイプライン中の自己同期型転送制御回路 C および C′ のそれぞれのリソース使用数を求めた．結果を表 2 に示す．Cyclone IV では Logic Cell が，Zynq-7000 では LUT と FF が，それぞれのリソースの単位である．Cyclone IV では，提案回路により Logic Cell 数が 50%削減された．

次に，スループットを評価するため，各パイプライン段においてパイプラインタクト（$T_{fi} + T_{ri}$）を静的タイミング解析により求めた．結果を図 9 に示す．パイプラインタクトは CAD ツールによる配置配線により増減し得る．図では，それぞれの FPGA において，提案回路におけるパイプラインタクトの最大値を 1 として正規化している．パイプラインの最大スループットは，最長となるパイプラインタクトの逆数である．Cyclone IV において，提案回路 C′ の最長のパイプラインタクトは従来回路 C に対して約 68%削減され，従って，スループット（= 1/ 最長パイプラインタクト）は，理想的には最大で約 3.2 倍に向上できることを確認した．

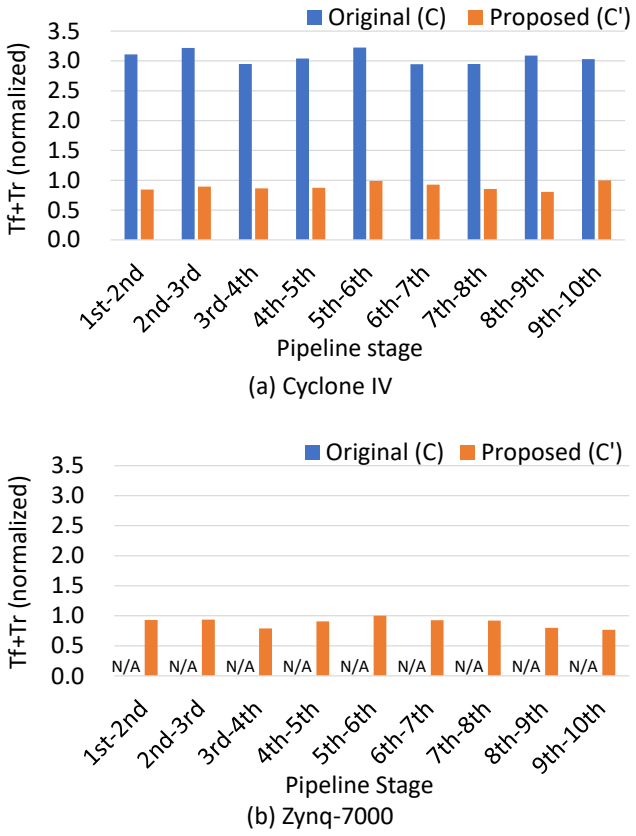

図 9　パイプライン基本構成におけるデータ転送時間

4.2　プロセッサ構成におけるスループット

提案回路を自己同期型データ駆動プロセッサに適用した場合のスループットを評価するため，ULP-CUE[5] から FL を取り除いた環状パイプラインを実装し，各パイプライン段においてパイプラインタクト（$T_{fi} + T_{ri}$）を静的タイミング解析により求めた．3.2 節で示したように，データの消去，複製，パイプラインの分流，合流機能を持つパイプライン段には，式 (3) に示したタイミング制約が存在する．各制約に対応する Dx に遅延を付加することで，制約を満足させることができるが，本稿では，タイミング制約を理想的に満足した状態のパイプラインタクトを評価する．具体的には，式 (3) の左辺が示すハンドシェイク信号の伝搬時間を，その下限となる式 (3) の右辺が示す機能制御信号の伝搬時間とした場合のパイプラインタクトを算出した．また，各 DL と FL に関わるレイテンシについては，これらのレイテンシがハンドシェイクにかかる時間内に収まるように理想的にパイプライン分割できた場合を想定し，Dx を取り除いた．結果を図 10 に示す．図 2 で示した ULP-CUE の MM，PS，FP，MA はそれぞれ 2 つ以上のパイプライン段で構成される．図 10 において，例えば MM0 は MM の 1 段目を表している．また，Cex はデータを消去するパイプライン段で用いる消去用 C を，Cce はデータの消去と複製を行うパイプライン段で用いる消去・複製用 C を，Cm はパイプラインの合流を実現するパイプ

ⓒ 2021 Information Processing Society of Japan

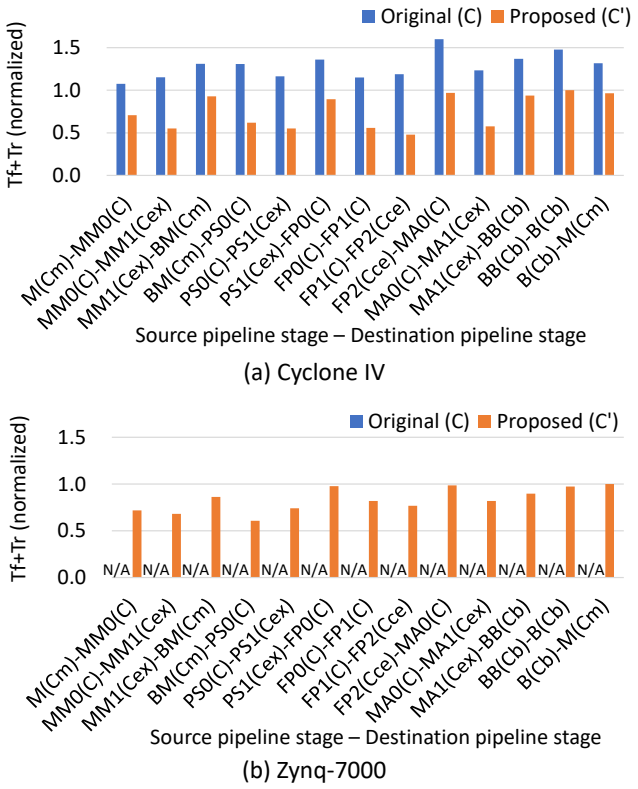

図 10　プロセッサ応用構成におけるデータ転送時間

ライン段で用いる合流用 C を表している．

Cyclone IV において，提案回路 C′ を用いた場合の最長パイプラインタクトは従来回路 C を用いた場合に比べて約 37%削減され，従ってスループットは，理想的には最大で約 1.6 倍に向上できることを確認した．

5. おわりに

本稿では，LUT を活用した小規模な自己同期型転送制御回路と，それに基づく自己同期型パイプラインの FPGA 実装を可能とする設計手順からなる回路構成法を提案した．Xilinx 社と Intel 社の FPGA を対象として，提案回路構成法に基づき自己同期型パイプラインを実装し，その結果に基づき，提案回路は従来に比べて 50%の小規模化を実現でき，これにより，パイプラインのスループットを最大で約 3.2 倍に向上できることに加え，応用例の一つであるプロセッサのスループットを最大で約 1.6 倍に向上できることを確認した．

提案回路構成法による高スループット化を最大限に活用するには，細粒度のパイプライン分割が欠かせない．具体的なプロセッサの細粒度パイプライン分割を実現した上で，そのスループットを定量的に評価することが今後の課題である．

参考文献

[1] H. Terada, S. Miyata, and M. Iwata, “DDMP’s: Self-Timed Super-Pipelined Data-Driven Multimedia Processors,” Proc. IEEE, Vol.87, pp.282–295, Feb. 1999.
[2] S. Yoshikawa, S. Sannomiya, M. Iwata, and H. Nishikawa. “Pipeline Stage Level Simulation Method for Self-Timed Data-Driven Processor on FPGA,” 2020 8th International Electrical Engineering Congress, Vol. 1, pp1–4, Mar. 2020.
[3] C. J. Myers, “Asynchronous circuit design,” Univ. of Utah John Wiley & Sons, Inc., 2001.
[4] 西川博昭，青木一浩，三宮秀次，宮城桂，岩田誠，宇津圭祐，石井啓之，“超低消費電力化データ駆動ネットワーキングシステムとその評価,” 電子情報通信学会論文誌 B, Vol.J96-B, No.6, pp.572-579, June 2013.
[5] 三宮秀次，青木一浩，宮城桂，岩田誠，西川博昭，“超低消費電力化データ駆動ネットワーキングプロセッサ ULP-CUE の試作とその評価,” 情報処理学会論文誌コンピューティングシステム, Vol.6 No.1. pp.78–86, Jan. 2013.
[6] M. Tranchero, and L. M. Reyneri, “Implementation of Self-Timed Circuits onto FPGAs Using Commercial Tools,” Proc. DSD, pp.373–380, Jan. 2008.
[7] J. Furushima, M. Nakajima, and H. Saito, “Design of an Asynchronous Processor with Bundled-data Implementation on a Commercial Field Programmable Gate Array,” Informatica, Vol. 40, pp.399–408, Nov. 2016.
[8] 三宮秀次, 大森洋一, 酒居敬一, 岩田誠, ”自己タイミング型パイプラインシステムの性能見積りモデル,” 電子情報通信学会論文誌 A, Vol.J92-A, No.7, pp.477-486, July 2009.

ⓒ 2021 Information Processing Society of Japan