FPGAを用いた機能回路コンポーネント間のPublish/Subscribe通信フレームワークの提案

新井 健太[1,a)]　大川 猛[1]　大津 金光[1]　横田 隆史[1]

概要：FPGA によるハードウェアアクセラレーションが幅広い分野で研究・開発され，FPGA 上に実装されるアプリケーションは複雑かつ大規模化してきている．また効率的に短期間で並列分散処理を実現するために，モデル駆動開発が広く用いられている．しかし回路設計においては，機能回路間のデータフローを信号レベルで記述しなければならず，開発生産性が低い問題がある．そこで本研究では，疎結合な分散システムのモデルである Publish/Subscribe 通信モデルを元に，機能回路間で Publish/Subscribe 通信を行う手法について検討した．本稿では，FPGA で Publish/Subscribe 通信を実現するための通信フレームワークについて述べる．

キーワード：FPGA，モデル駆動開発，Publish/Subscribe 通信

1. はじめに

Publish/Subscribe 通信モデル [1] は，今日のインターネットを活用した情報処理アプリケーションのデータフローを表現するモデルとして有用性が広く認識されている [2][3][4]．すなわち，多数の Publisher が発信した情報のうち，必要なトピックだけを Subscriber が受信するというモデルは，ビッグデータの時代における情報処理の自然な構造を表している．例えば ROS(Robot Operating System)[5] は，ロボットを動かすための多くのソフトウェアプロセスの間でなされるデータ通信を Publish/Subscribe の通信モデルで設計・実装することにより，ロボットのソフトウェア部品のスケーラビリティ・拡張性・再利用性を向上した．

一方，FPGA(Field Programmable Gate Array) はエネルギー効率の高い処理プラットフォームとして活用が期待されている．しかし回路設計の生産性が低く，多くの場面においてシステムへの導入が困難であるという問題がある．これは現行の FPGA 開発環境が設計資産 IP(回路モジュール) の接続を電気信号レベルで行っており，回路モジュールの可搬性 (ポータビリティ) が低いことに起因していると考えられる．

そこで本研究では，FPGA 上に Publish/Subscribe 通信モデルに基づく並列分散システムを設計する手法について検討した．本稿では FPGA 上に Publish/Subscribe 通信を実現する手法について述べる．

[1] 宇都宮大学大学院工学研究科
Graduate School of Engineering Utsunomiya University, Yoto 7–1–2, Utsunomiya, Tochigi, 321–8585, Japan
[a)] kenta@virgo.is.utsunomiya-u.ac.jp

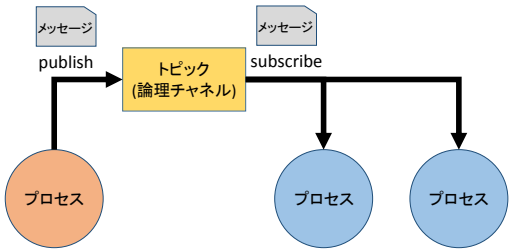

図 1　Publish/Subscribe 通信モデル

2. Publish/Subscribe 通信モデル

2.1 Publish/Subscribe 通信モデルの概要

Publish/Subscribe 通信モデルにおける通信の形態を図 1 に示す．このモデルにおいて，各プロセス (処理を実行する機能の単位) は他のプロセスとトピック (論理チャネル) を介して通信を行う．Publisher(送信側のプロセス) が送信するデータはトピックによって分類する．また Subscriber(受信側のプロセス) は必要なデータを配信するトピックのみからデータを受信する．トピックに接続する Publisher と Subscriber はそれぞれ複数存在してよい．

2.2 並列処理システムの設計

Publish/Subscribe 通信を利用した並列処理システムの有利な点は，既存のバスプロトコルと比較して，プロセスの追加や削除が容易なことである．例えば既存のバスプロトコルである AMBA AXI バス [6] はメモリマップ方式のプロトコルであり，図 2 の (a) に示すように，マスタがスレーブのアドレスを指定してデータの読み書きを行う [7]．

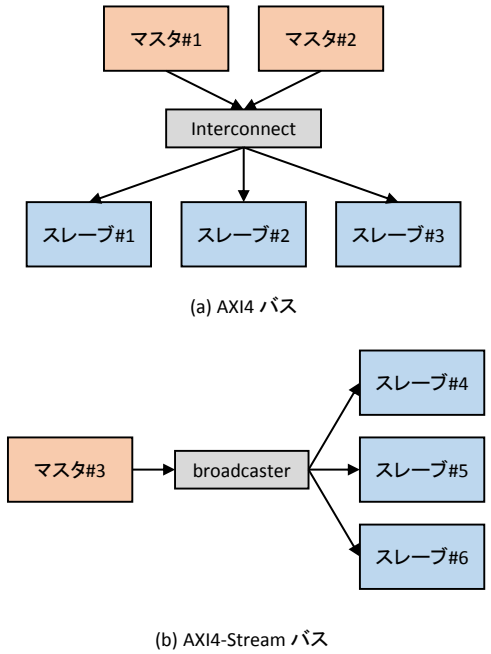

図 2　AMBA AXI バス

したがってスレーブを追加する場合，各マスタがアクセスするアドレスを追加する必要がある．また，AXI4-Stream プロトコルはストリームデータを送受信するためのメモリアドレスが不要なプロトコルであり，複数のプロセスに対してブロードキャストすることも可能である [8]．しかしマスタはスレーブが受信可能な状態であることを伝える信号 TREADY を待つ必要がある．対して Publish/Subscribe 通信モデルでは，マスタとワーカはトピックに対してデータの送受信をするために，お互いの状態を知る必要がない．そのため，要求に応じたシステムの変更がしやすい．

Publish/Subscribe 通信モデルに基づいてシステムを設計する場合，図 3 に示すように，複数のプロセスを繋ぎ合わせることでシステムを実現する．これらのプロセスはトピックから受信したデータを処理し，随時別のトピックに送信する．このシステムにおいて各プロセスは独立して動作するため，処理のパイプライン化が可能となる．また個々のプロセスについても，データ並列性を活用した高速化が可能であれば，プロセス数を増やして並列化することも可能である．

3. Publish/Subscribe 通信フレームワーク

これまでの議論では，Publish/Subscribe 通信モデルに基づいて並列処理システムを設計することで，機能の追加や削除が容易になることを述べた．本節では，FPGA 上の機能回路間で Publish/Subscribe 通信を実現するためのフレームワークについて述べる．

3.1 フレームワークへの要求

FPGA 上の機能回路間で Publish/Subscribe 通信を実現

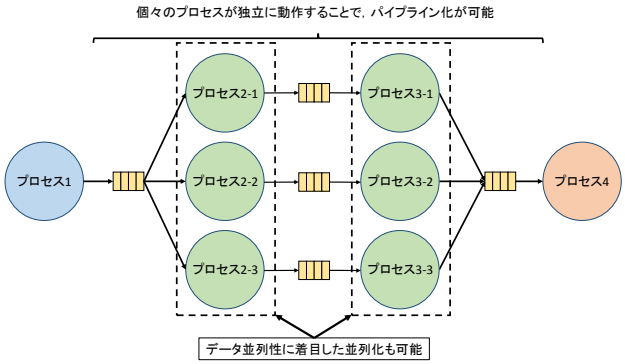

図 3　Publish/Subscribe 通信モデルに基づく並列処理システム

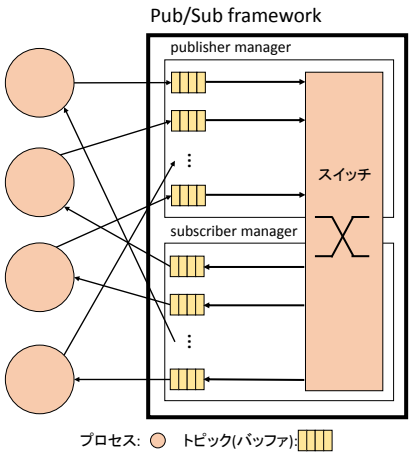

図 4　Publish/Subscribe 通信のためのフレームワーク

するために，フレームワークに対する要求を以下に示す．

(1) Publisher がトピックに送信したデータを Subscriber に配信する

(2) 複数の Publisher，Subscriber に対応する

Publish/Subscribe 通信において，送受信するデータはトピックを介して配信される．そのため，(1) で示すように，本フレームワークは Publisher と Subscriber を仲介する必要がある．また Publish/Subscribe 通信では，送信者と受信者が共に複数存在する可能性がある．したがって (2) に示すように，本フレームワークも複数の Publisher と Subscriber に対応させる．

3.2 フレームワークの機能

実現するフレームワークのモデルを図 4 に示す．前小節の要求を満たすために実装する機能は以下の 4 点である．

(1) 機能回路が送信したデータを保存する FIFO バッファ

(2) 機能回路が受信するデータを保存する FIFO バッファ

(3) トピックごとに FIFO バッファへデータを振り分ける

このフレームワークにおいて，トピックは FIFO バッファに割り当てる．そして Publisher が入力したデータを，Subscriber と接続する FIFO バッファに出力する．同じトピックに対応する FIFO バッファの数を制限しないことで，機能回路間で N 対 M の通信を可能にする．

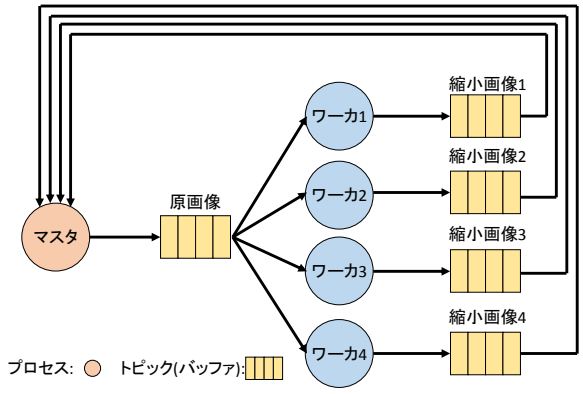

図 5　画像縮小システムの Publish/Subscribe 通信ネットワーク

4. フレームワークを用いた設計事例

4.1 設計する並列処理システム

前述のフレームワークを使用した設計事例として，画像の縮小処理を異なる倍率で並列に行うシステムの Publish/Subscribe 通信モデルを説明する．このような異なる倍率の画像縮小処理は，例えば局所特徴量を使用する画像認識処理において用いられる．

Publish/Subscribe 通信モデルに基づく縮小画像の生成処理モデルを図 5 に示す．マスタは原画像をワーカに配信する．各ワーカは受信した画像から縮小画像を生成する．

4.2 システム構成

設計した画像縮小システムを図 6 に示す．フレームワークには Publish 用の FIFO バッファを 5 つ，Subscribe 用の FIFO バッファを 8 つ使用した．トピック 1 は原画像用のチャネルで，Publisher はマスタ，Subscriber はワーカである．トピック 2 から 5 は縮小画像用のトピックで，Publisher は各ワーカ 1 つ，Subscriber はマスタである．マスタはトピック 1 を介して全ワーカに原画像を配布する．そして各ワーカは原画像を縮小し，トピック 2 から 4 に縮小画像を書き込む．

5. 評価

提案したフレームワークのスイッチ機能を実装し，ハードウェア使用量と動作周波数について評価した．フレームワークの実装は，Xilinx 社の Vivado HLS 上で C++言語を用いて行った．

ソースコードを図 7 に示す．pubsub_framework 関数は，前節で述べた画像縮小処理用の Publish/Subscribe 通信用フレームワークである．ap_uint<8>型は 8 ビットのデータを表現する．また hls::stream<>型はストリーム入出力用のテンプレート型であり，合成後は FIFO として扱うことができる．したがって、この関数の引数一つ一つが、それぞれ FIFO に対応する．ptopic0 から 4 は Publisher 用，stopic1 から 4, 01 から 04 は Subscriber 用のインター

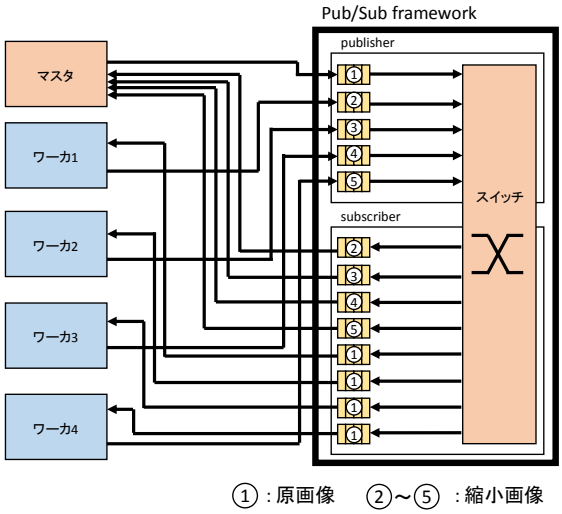

図 6　フレームワークを用いた画像縮小システムのブロック図

フェースである．これらの FIFO インターフェースに対し，37 から 58 行目でスイッチ処理を行ってデータを振り分けている．

図 7 のソースコードを Vivado HLS で合成し，ハードウェア使用量と推定される動作周波数を得た．なお実装する FPGA は Digilent 社の Genesys2 ボード [9] に搭載されている XC7K325T–2FFG900C とした．ハードウェア使用量を図 8 に示す．なおこのハードウェア使用量には FIFO バッファは含まれていない．また推定される動作周期を図 9 に示す．本フレームワークはレジスタが 2 つ，LUT が 134 個で構成できた．これは Genesys2 ボード上のレジスタ，LUT の 1%を下回るリソース量である．また動作周期は 4.38ns，Uncertainty が 0.62 であることから最大でも 5ns に収まることが分かった．すなわち実装したフレームワークは最大で 200MHz での動作が可能である．

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	4
FIFO	-	-	-	-
Instance	-	-	-	-
Memory	-	-	-	-
Multiplexer	-	-	-	130
Register	-	-	2	-
Total	0	0	2	134
Available	890	840	407600	203800
Utilization (%)	0	0	~0	~0

図 8　ハードウェア使用量

6. おわりに

本稿では，Publish/Subscribe 通信モデルに基づく機能回

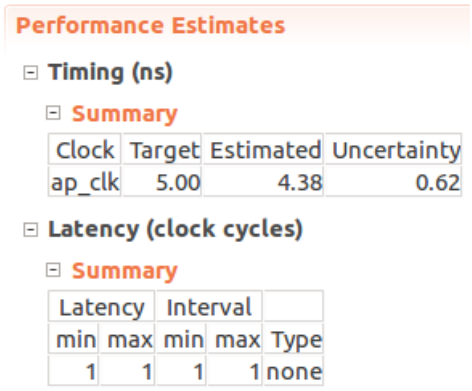

図 9　推定される動作周期

路間の通信用フレームワークについて述べた．本フレームワークは，トピックを FPGA チップ上の FIFO バッファに割り当て，また Publisher，Subscriber の数だけ FIFO バッファを用意することで，Publish/Subscribe 通信のためのインターフェースを構成する．

今後は画像処理システムを実装し，処理時間とハードウェア使用量について評価を行う予定である．

謝辞　本研究は一部 JSPS 科研費 15K00068，16K00068，17K00072 の助成による．

参考文献

[1] P. T. Eugster, P. A. Felber, R. Guerraoui, and A. M. Kermarrec, ”The Many Faces of Publish/Subscribe”, ACM Computing Surveys, vol.35, no.2, pp.114–131, June 2003.

[2] D. Lagutin, K. Visala, and S. Tarkoma, ”Publish/Subscribe for Internet: PSIRP Perspective,” Towards the Future Internet Emerging Trends from European Research, 2010.

[3] K. H. Tsoi, I. Papagiannis, M. Migliavacca, W. Luk and P. Pietzuch, ”Accelerating publish/subscribe matching on reconfigurable supercomputing platforms,” Many-Core and Reconfigurable Supercomputing Conference (MRSC), Rome, Italy, Vol.3, 2010.

[4] Abhishek Mitra ,Marcos R. Vieira, Petko Bakalov, Walid A. Najjar, and Vassilis J. Tsotras, ”Boosting XML filtering through a scalable FPGA-based architecture,” CIDR, 2009.

[5] ROS offitial page, https://www.ros.org, アクセス日: 2018/7/5.

[6] AMBA AXI specification, https://www.arm.com/products/system-ip/amba-specifications, アクセス日: 2018/7/5.

[7] Xilinx Inc., ”AXI Reference Guide UG761 (v13.1),” https://www.xilinx.com/support/documentation/ip_documentation/ug761_axi_reference_guide.pdf, アクセス日: 2018/7/5.

[8] Xilinx Inc., ”AXI4-Stream Infrastructure IP Suite v2.2 PG085,” https://www.xilinx.com/support/documentation/ip_documentation/axis_infrastructure_ip_suite/v1_1/pg085-axi4stream-infrastructure.pdf, アクセス日: 2018/7/5.

[9] Digilent Inc, ”Genesys 2 Kintex-7 FPGA Development Board”, https://store.digilentinc.com/genesys-2-kintex-7-fpga-development-board/, アクセス日: 2018/7/5.

```
1  #include "ap_int.h"
2  #include "hls_stream.h"
3
4  ap_uint<8> buf;
5
6  void pubsub_framework(
7      // topic for publisher
8      hls::stream< ap_uint<8> > &ptopic0,
9      hls::stream< ap_uint<8> > &ptopic1,
10     hls::stream< ap_uint<8> > &ptopic2,
11     hls::stream< ap_uint<8> > &ptopic3,
12     hls::stream< ap_uint<8> > &ptopic4,
13     // topic for subscriber
14     hls::stream< ap_uint<8> > &stopic1,
15     hls::stream< ap_uint<8> > &stopic2,
16     hls::stream< ap_uint<8> > &stopic3,
17     hls::stream< ap_uint<8> > &stopic4,
18     hls::stream< ap_uint<8> > &stopic01,
19     hls::stream< ap_uint<8> > &stopic02,
20     hls::stream< ap_uint<8> > &stopic03,
21     hls::stream< ap_uint<8> > &stopic04
22 ){
23 #pragma HLS INTERFACE ap_ctrl_none port=return
24 #pragma HLS INTERFACE ap_fifo port=ptopic0
25 #pragma HLS INTERFACE ap_fifo port=ptopic1
26 #pragma HLS INTERFACE ap_fifo port=ptopic2
27 #pragma HLS INTERFACE ap_fifo port=ptopic3
28 #pragma HLS INTERFACE ap_fifo port=ptopic4
29 #pragma HLS INTERFACE ap_fifo port=stopic1
30 #pragma HLS INTERFACE ap_fifo port=stopic2
31 #pragma HLS INTERFACE ap_fifo port=stopic3
32 #pragma HLS INTERFACE ap_fifo port=stopic4
33 #pragma HLS INTERFACE ap_fifo port=stopic01
34 #pragma HLS INTERFACE ap_fifo port=stopic02
35 #pragma HLS INTERFACE ap_fifo port=stopic03
36 #pragma HLS INTERFACE ap_fifo port=stopic04
37     // topic 0
38     if(!ptopic0.empty()){
39         buf = ptopic0.read();
40
41         stopic00.write(buf);
42         stopic01.write(buf);
43         stopic02.write(buf);
44         stopic03.write(buf);
45     }
46     // topic 1
47     if(!ptopic1.empty())
48         stopic1.write(ptopic1.read());
49     // topic 2
50     if(!ptopic2.empty())
51         stopic2.write(ptopic2.read());
52     // topic 3
53     if(!ptopic3.empty())
54         stopic3.write(ptopic3.read());
55     // topic 4
56     if(!ptopic3.empty())
57         stopic4.write(ptopic4.read());
58 }
```

図 7　Publish/Subscribe 通信用フレームワークのソースコード