東京工業大学工学部

学士論文

FPGAで実現するコンピュータシステムの提案とその初期検討

指導教員 吉瀬 謙二 准教授

平成27年2月

提出者

学科　　情報工学科
学籍番号 11_27843
氏名　　味曽野 智礼

指導教員認定印		
学科長認定印		

平成26年度　学士論文内容梗概

FPGAで実現するコンピュータシステムの提案とその初期検討

指導教員　　吉瀬 謙二　准教授
情報工学科
11_27843　　味曽野 智礼

FPGA の高性能化に伴い、従来ASIC で作られてきた複雑な回路も FPGA に実装されてきている。FPGA の性能向上は今も続いており、今後更に多くの分野においてその利用が広がると予想される。また近年では数値計算や画像処理などの特定アプリケーションのアクセラレータの開発が進んでおり、FPGAに実装されているものも存在している。これらの多様なアクセラレータをコンピュータシステムに接続し、利用することが出来ればシステム全体の性能を向上させることが出来ると考えられる。しかし従来システムの内部接続に用いられたバスネットワークにはスケーラビリティが低いという問題があり、多くの構成要素を接続することが難しい。

これらの背景より、本論文ではCPU やアクセラレータを FPGA に実装し、それらやメモリをメッシュネットワークに接続した拡張性、スケーラビリティの高いコンピュータシステムを提案する。また提案システムの初期検討としてそのメモリ性能を、動作周波数とルータ間の通信速度に着目してシミュレータで測定した。その結果、2015年時点で実現できる FPGA 性能ではルータ間の通信速度がボトルネックとなり、メッシュネットワークによる性能はバスネットワークと比べて大きく低下することがあることがわかった。

目　次

第1章

序論

FPGA の高性能化に伴い、従来ASIC で作られてきた複雑な回路も FPGA 上に実装されてきている。FPGA は論理合成を通じて回路をプログラミングするため、回路の修正や更新が可能であり、低コスト/短期間で柔軟な開発が行える。FPGA の性能向上は今も続いており、現在数百 MHz 程度が主流であるFPGA の動作周波数は将来的に1GHz を超え、更により多くの分野において利用されると予想される。

また、近年では数値計算や画像処理などの特定アプリケーションのアクセラレータの開発が行われている。コンピュータシステムが必要となる処理に応じたアクセラレータを多数持つことにより、システム全体の高速化が可能になると考えられる。

しかしながら、従来のコンピュータシステムを構成するバスネットワークは少数の接続の場合に高速で動作するものの、多数の構成要素を接続することには適していない。これは接続数が増えるに連れて通信の衝突が多発し、調停機構の複雑さが増すため、接続数に性能がスケールしにくいという問題点があるためである。接続数にスケールさせる一つの方法は、メッシュネットワークなどの一つのノードの接続数が決まっている接続方法を用いることである。

これらの背景より、 将来的にはCPU やアクセラレータを FPGA に実装し、それらやメモリをメッシュ状のネットワークに接続することで拡張性、スケーラビリティの高いコンピュータシステムを構築することができると考える。

本論文ではそのようなFPGAで実現するコンピュータシステムの構成を提案し、その検証モデルの設計を示す。またシステムの初期検討として、検証モデルのメモリシステム性能が、従来のバス接続ネットワークに比べてどの程度であるのかを、シミュレーションの結果から議論する。

本論文の構成

本論文の構成は以下の通りである。まず2章で研究背景となる要因を述べ、続く3章で提案するコンピュータシステムの概要を説明する。4章ではシステムの初期検討として、検証モデルにおけるメモリ性能を従来のバス接続の場合と比較する。5章では4章の結果をもとに考察を行い、最後に6章で結論を述べる。

第2章

研究の背景

この章では提案する計算機システムの背景とそのシステムの概要について説明する。

研究の背景には大きく分けて3つの要因が存在する。

2.1 バス接続のスケーラビリティの問題とNoC[4][1]

従来の計算機はバスネットワークにより CPU やメモリといった部品が接続されている。より最近のチップセットにおいては単純なバスではなく、リングバスが用いられている。

バスネットワークには、接続部品が少ない場合にコントローラの設計が簡潔んになり、高速に動作するという利点がある。しかし1度に1つの要求にしか応じられないためバスのバンド幅は小さく、また接続部品が増えるに連れて調停機構が複雑になるため、接続数にスケールした性能を出すことが難しくなる。

一方で、NoC (Network On Chip) とはパケット交換により通信を行う接続プラットフォームであり、ノード (プロセッサやメモリなどのリソースとルータの組) が相互に接続される。

NoC ではバスネットワークと違い任意の数の接続に性能がスケールする。この理由は、1つのルータの入出力数の上限がトポロジーにより一意に決まるからである。例えば2次元メッシュネットワークにおいては1つのルータに最大4入力であり、これはシステム全体のノード数が増えても変わらないため、1つのノードが1度に処理しなければならない要求の数は変化しない。また NoC では複数のノードの組が同時に通信を行うことができるため、システム全体のバンド幅が高くなる。一方で、NoC ではパケット交換のために一定のレイテンシが発生することが短所となる。

2.2 FPGA 利用の広がり

FPGA (Field Programmable Gate Array) は、Verilog や VHDL などの HDL (Hardware Description Language : ハードウェア記述言語) により構成を記述し、論理合成を行うことで回路の書き換えを行うことができるロジック・デバイスである。

従来の ASIC (Application Specific IC) による電子部品の開発では、基盤が実装された後に回路を変更することが出来ない。一般に ASIC の開発には数年かかり、コストが高い。また製品が出来上がるまでの時間が長いため、先進的なアーキテクチャの設計をすぐに利用することが難しい。

一方で FPGA の性能は ASIC に劣るが、開発コストや開発にかかる時間は小さく、通信処理などの特定の分野においては ASIC の代わりに用いられるようになってきている。論理合成の書き換えを通じて、回路の修正や更新が可能なことも FPGA の利点である。

テクノロジーの進化により、FPGA はより高性能、低消費電力、低コストになっている。現在の FPGA の動作周波数は最高数百 MHz 程度であるが、将来的には 1GHz 以上の動作周波数が実現されると予想される。

2.3 アクセラレータの利用による計算能力の向上

コンピュータ・システムがあらゆる分野に使われるようになるにつれ、汎用的な計算処理を行う CPU だけでなく、特定の計算を高速に行うアクセラレータの利用が増えてきている。典型的な例としては画像処理に特化して並列計算を行う GPU が上げられる。CPU は高速なスカラ計算に重点を置いてるため、効率よく並列にデータを処理することが得意ではない。GPU を用いて並列計算処理を行うことで、CPU では行うことの出来ない高速化を達成することができる。

現在では GPU の並列処理能力は画像処理以外にも利用されるようになり、それらは GP-GPU (General Purpose GPU) と呼ばれている。しかし、アプリケーションの中には GP-GPU の利用だけでは潜在的高速化を最大限に引き出すことが難しいものもある。もしも処理能力の高い FPGA を用いて柔軟にアクセラレータを用意することが可能ならば、CPU 単体や GP-GPU を利用した場合よりもプログラム処理を高速化することが可能になると考えられる。

第3章

FPGAで実現する コンピュータシステムの提案

3.1 提案するコンピュータシステム[6]

前節の内容をまとめると以下のようになる。

- ASIC の性能は高いが、数年単位の開発が必要でありコストが高い
- FPGA は柔軟な開発が可能であり、FPGA 自体の性能向上は今後も見込まれる
- CPU に加えてアクセラレータを使用することで、処理の高速化を行える
- 従来のコンピュータシステムで用いられているバス接続はスケーラビリティが低い
- バスに代わる接続として NoC (2次元メッシュ) が考えられる

以上の背景を踏まえて、我々はFPGAを用いた、新しいコンピュータシステムを提案する。この提案システムにおいてはプロセッサコアや各種アクセラレータはFPGAに実装され、それらやメモリなどが2次元メッシュネットワークにより接続される。ユーザーの要求(動かしたいプログラム)によりシステムの構成(メモリやアクセラレータの数/種類など)を柔軟に変更することのできる、拡張性とスケーラビリティの高いコンピュータシステムの実現を目標とする。

3.2 検証モデルの構成

前章で説明したコンピュータシステムの評価を簡略化したモデルで行うことを考える。検証モデルとして以下の構成を考える。

1. プロセッサコア, サブコア (アクセラレータ), I/O ポート, メモリ の 4 つを主要な構成部品 (ノード) とする
2. 各構成部品は 2 次元メッシュネットワークにより接続される
3. OS が動作するプロセッサコアはシステム上にただ 1 つとする
4. メモリやアクセラレータは任意の場所に任意の数接続できる
5. サブコア (アクセラレータ) はメインコアからプログラムを割り当てられ、それを実行する
6. サブコア (アクセラレータ) でのスレッド切り替えは考慮しない
7. メインコア/各サブコアで動くプログラムはそれぞれ独立である (各プログラムの使用するメモリ領域は独立であり、コヒーレンス制御を考えない)
8. メモリ/サブコアの接続位置/関係は制限しない
9. ルーティング経路はただ 1 つに決まる
10. コアからのメモリアクセス要求は高々1 つしか同時に起こらない

このシステムの使用モデルは、メインコア上で動作する OS が実行するプログラムに応じて順次、適切に処理をサブコアへ割り当てる (メモリ領域の確保/プログラムの転送を行う) というものである。求めるシステム全体の処理能力に応じて、メモリやサブコアの構成/数を配置する (図 (3.1))。

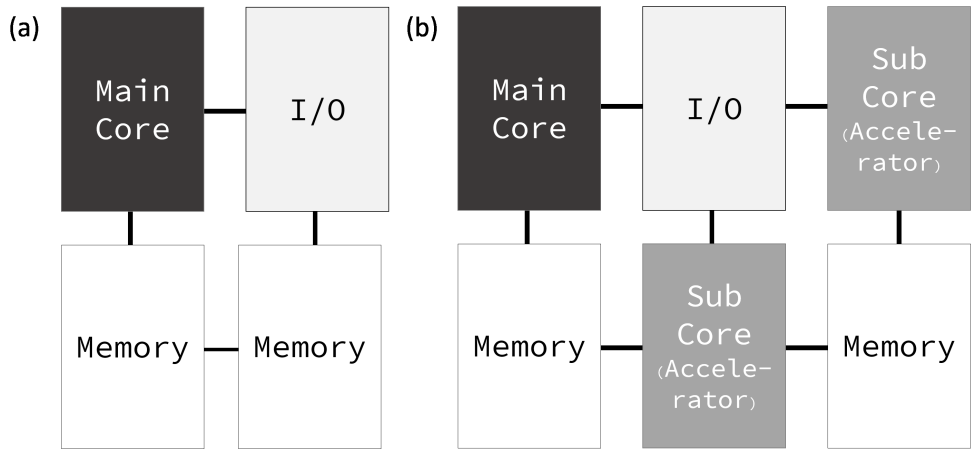

図 3.1: 検証モデルの構成例

第4章

初期検討

この章では前章で示した検証モデルのメモリ性能についてシミュレーションを用いて簡単な評価を行った結果を示す。

具体的には各サブコアが同時に処理を開始した時の性能 (実行時間) を、バス接続にした場合とメッシュ接続にした場合で比較した。コアの総数は 8 以下とした。今回は OS の処理やストレージからのプログラムロードの時間については考慮しておらず、サブコアとメモリの間の通信のみに焦点を当てている。

評価環境にはシステムシミュレータの gem5 を使用した。

4.1 シミュレーション環境： gem5

サイクルアキュレートなシミュレーションを行うことができる gem5[2] を利用した。

4.1.1 gem5 の概要

gem5 は M5[3] と GEMS[5] という 2 つのシミュレータを組み合わせて開発され、特にシミュレータコアの部分は M5, メモリ階層シミュレーション部分は GEMS (の Ruby) を元にしている。gem5 の開発は アメリカの複数の大学や企業 (ウィスコンシン大学メディスン校/MIT/ミシガン大学,ARM/AMD/HP など) が共同で行っている。

gem5 では Linux などの OS を起動する事ができるフルシステムモード (FS モード) と、スタティックリンクされた Linux ELF バイナリ単体を実行するシステムエミュレーションモード (SE モード) が存在し、今回はを SE モードを使用した。SE モードでもシステムコールを使うことが可能である。SE モードでは複数コアに独立したプログラムを割り当てて実行することが可能であり、この時はどれか 1 つのプログラムが終了した時点でシミュレーションが終了する。

gem5 のシミュレータ本体は C++ で書かれているが、python インタプリタがリンクされており、シミュレータの設定は python で記述された設定ファイル

を用いて行う。実行時に与える python のシステム設定ファイルを変更することで、シミュレーション対象のハードウェア構成を変更することが可能である。

なお、シミュレーションはシングルスレッドで動作するためにマルチコア環境のシミュレーションにおいてはコア数が増加するにつれてシミュレーション時間が線形的に増大する。

4.1.2 gem5 のコンフィギュレーション

当然ながら gem5 のシステム構成はバスを全体に作られている。今回は gem5 本体の仕様を変更すること無く、システム設定ファイルを変更することのみで、擬似的にメッシュ接続を再現した。これは、検証モデルにおいて 1 つのコアが出せるメモリ要求が高々1 つだけである (あるメモリ要求が完了するまで、次のメモリ要求は出せない) という仮定のもとで成り立つ。

4.2 メッシュ接続の擬似的な再現

4.2.1 1 つのコアが 1 つのメモリを使用する場合

図 (4.2) は 3 つのコアと 1 つのメモリノードが接続されている例を示している。図中の R はルーターを表している。この時、各コアのメモリへのアクセス経路は 図 (4.2) に示すとおりになる。メモリアクセスまでに Core0 と Core2 は 1 ホップ、 Core1 は 2 ホップ必要となることが分かる。

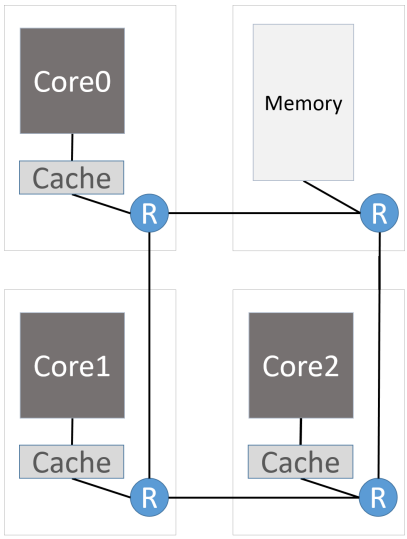

図 4.1: 3 コア+1 メモリの構成

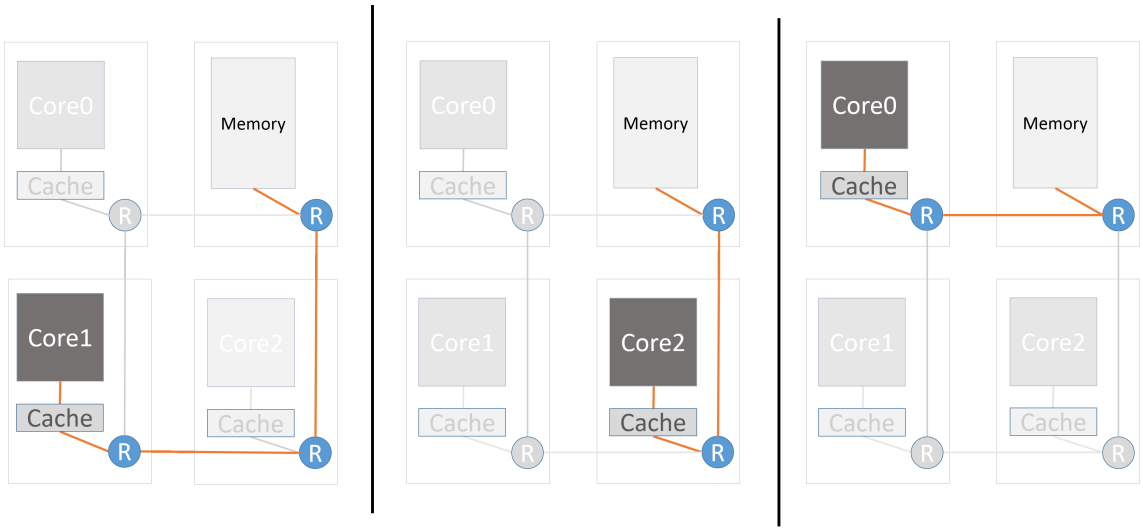

図 4.2: 各コアのメモリへのアクセス経路

ところでデフォルトでは gem5 で 1CPU とメモリの接続は 図 (4.3) の形になっている。

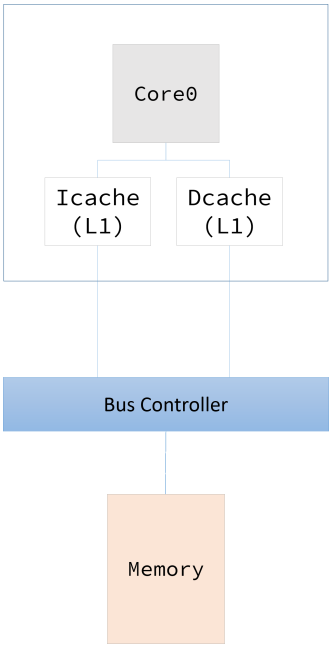

図 4.3: gem5 での 1CPU の接続

今回は gem5 においてメモリ直前のバスの前にダミーキャッシュ(ダミーレイヤー) をはさむことによりルーターの構造を簡易的に再現した (図 (4.4))。図中では 1 ホップに 25ns かかるとしている。ダミーキャッシュはサイズが L1 のキャッ

シュラインサイズと同じであり、ほぼ確実にキャッシュミスをする (キャッシュは内包でないので100%ではないが、ミス率は1%程度であるため今回は無視した)。各コアからのメモリ要求は同時に1つしか起こらないと仮定したため、ホップする時間をそのままダミーキャッシュへのアクセス/レスポンス時間へ割り当てて考えることとした。

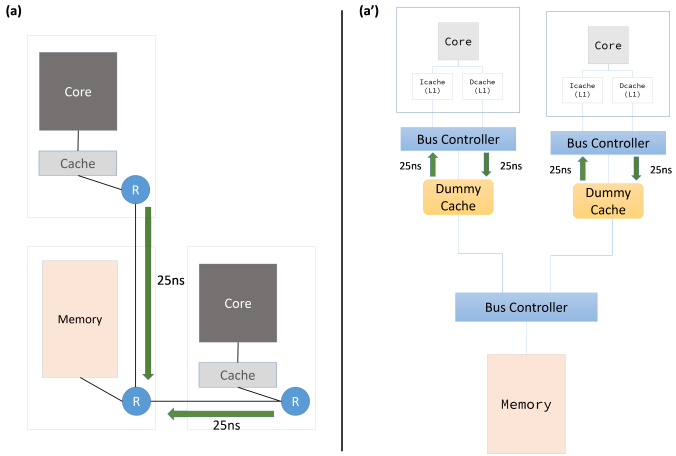

図 4.4: (a) ルータ間通信, (a') 対応するバス接続 (1 コア)

3つのコアを1つのメモリに接続する時は図 (4.5) のようになり、ホップ数に応じたダミーキャッシュをそれぞれのコアに用意する。

4.2.2 1つのコアが複数のメモリを使用する場合

1つのコアが複数のメモリを使う場合を考える。

コア:メモリが2:2であり、片方のコア (Core1) が1つのメモリだけ、もう片方のコア (Core0) は両方のメモリを使うとする。このとき 図 (4.6) (a) に示すように各コアがそれぞれ同時に別のメモリにアクセスしている時と、(b) のように同時に同じメモリにアクセスしている時が発生する。仮にメモリアクセスパターンが均等であるとすると、これは コア:メモリ が 1:1 の時と 2:1 の時にそれぞれ対応し、その割合が分かれば全体の性能を予測することができる。よってメッシュネットワークに接続されるコアの数が増えた場合も、極端に遠いメモリを使うことはないと考えれば、全体の性能を考える際もはある適度局所的なコアとメモリの接続関係を考慮すれば良いことがわかる。

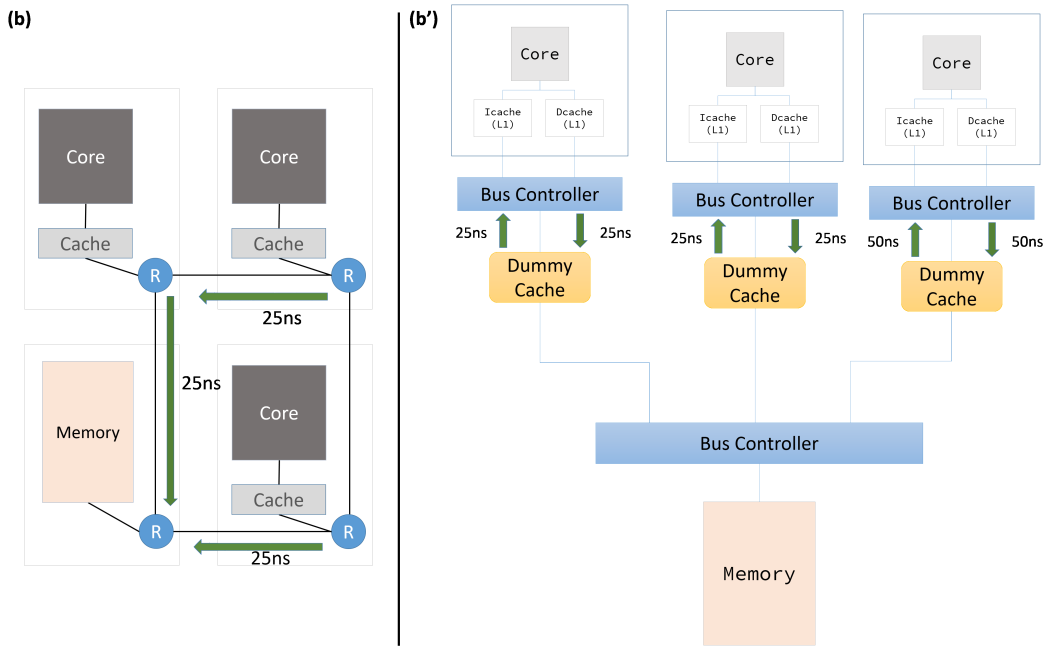

図 4.5: (b) ルータ間通信, (b') 対応するバス接続 (3 コア)

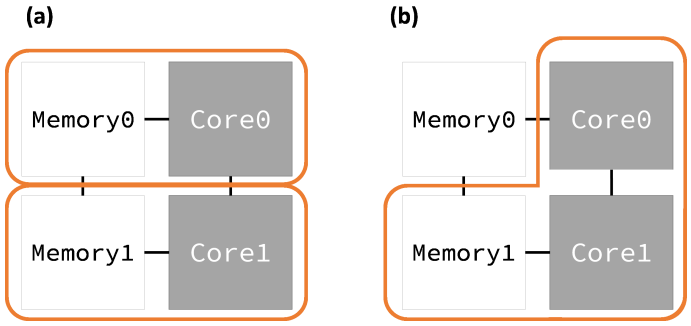

図 4.6: 1つのコアが複数のメモリを使うパターン：(a) 各コアが別のメモリにアクセスするとき (b) 2つのコアが同時に1つのメモリへアクセスするとき

4.3 評価対象

4.3.1 評価するネットワーク構造

今回は3×3の2次元メッシュ接続に現れるであろう コア:メモリ = N:1 のパターンについて考慮した。パターンとその略称は図 (4.7) に示すとおりであり、M,C のブロックがそれぞれメモリ、コアに対応している。ただし、3×3 のメッシュ構成においてメモリが右上端にあるなどの極端な構成は考えていない。図に示すようにコアからメモリへのアクセスまでは高々2ホップである。

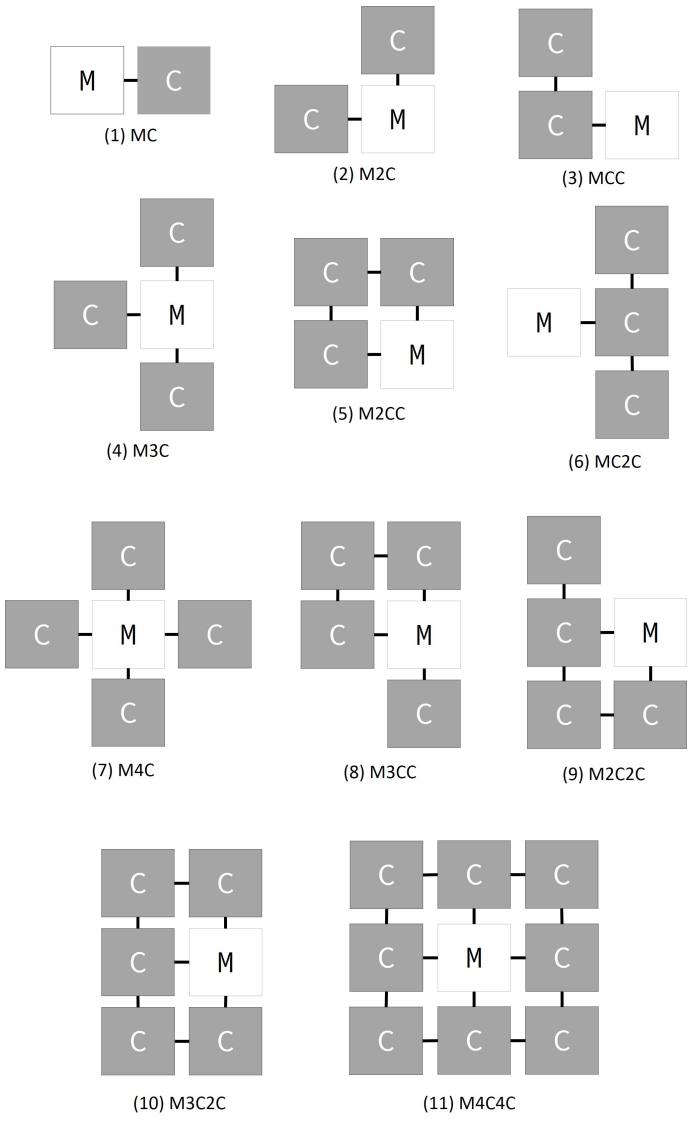

図 4.7: コア:メモリ = N:1 の 11種のパターン

4.3.2 評価対象の種類パラメータ

シミュレーションするシステムの性能は、2015 年現在で実現可能であろう FPGA 性能レベル (Real-Real, RR: 動作周波数 100MHz, 1hop = 320ns) と、将来 FPGA の性能が上がることを見越した上でのレベル (Ideal-Ideal, II: 動作周波数 1GHz, 1hop=25ns), 及び 動作周波数は現在のままだが通信速度は将来のレベルであるもの (Real-Ideal, RI: 動作周波数 100MHz, 1hop=25ns) の 3 つを用いた。ここで、1hop にはルーターの動作に 20〜30 サイクル程度かかると仮定している。3 つの評価対象の異なるパラメータ値を表 (4.1), 共通するパラメータを表 (4.2) に示す。

表 4.1: 評価対象により異なるパラメータ

	Real-Real	Real-Ideal	Ideal-Ideal
動作周波数	100MHz	100MHz	1GHz
メモリバンド幅	400MBps	400MBps	3200MBps
1 ホップ	320ns	25ns	25ns

表 4.2: 評価対象で共通するパラメータ

プロセッサ	x86-64 (OoO)
L1 キャッシュ	命令/データ 各 16kB
L2 キャッシュ	なし
キャッシュラインサイズ	8B
メモリサイズ	512MB

4.4 測定方法/実行プログラム

gem5 のシステムエミュレーションモードにおいて、各コアに独立に、全く同じプログラムを動かした時の実行時間を測定した。測定プログラムには 碁のシミュレーションを行うプログラムを用いた。実行される総命令数は 161,371,638 命令 (254,113,806 マイクロオペレーション) であり、マイクロオペレーションの内訳は表 (4.3) に示すとおりである。また、総命令フェッチ数は 約 330,000,000 命令であった (実行サイクル数に依存) する。

先述の通り gem5 では 1 つのコアでの実行が終了した時点でシミュレーショ

ンが終了するため、終了後出力される統計情報をもとに、実行が終了していなかったコアは負荷がそれまでの平均と同程度であると仮定して予測終了時間を求めた。

また、従来の通りにバス接続で1,2,4,8コアを接続して同時にプログラムを動かした場合の結果も測定した。

表 4.3: 測定プログラムの命令(マイクロオペレーション)内訳

タイプ	コミットされた命令数 (micro ops)	割合
Int ALU	160940238	63.33%
Int Mult	126961	0.05%
Read	70039425	27.56%
Write	22998728	9.05%
その他	8454	0.00%
合計	254113806	100.00%

4.5 シミュレーション結果

4.5.1 バスネットワーク接続

まずはじめに従来のバスネットワークによる接続での結果を表 (4.4) に、そのグラフを図 (4.8) に示す。バス接続の場合にはルータ間の通信速度は関係無いため、RR = RI の実行時間となる。この時各コアの動作はほぼ同時に進行し、1つのコアが動作を終了した時の他コアのプログラム実行率は99%以上である。コア数が 8 つの時は 1つの時の接続に比べて RR の場合は 約2.2倍, II の場合は 約1.8倍の実行時間となっており、バス調停に時間がかかっていることが分かる。

表 4.4: バス接続の予測平均実行時間 (s)

コア数	RR (RI)	II
1	2.336	0.350
2	2.450	0.357
4	2.912	0.389
8	5.074	0.632

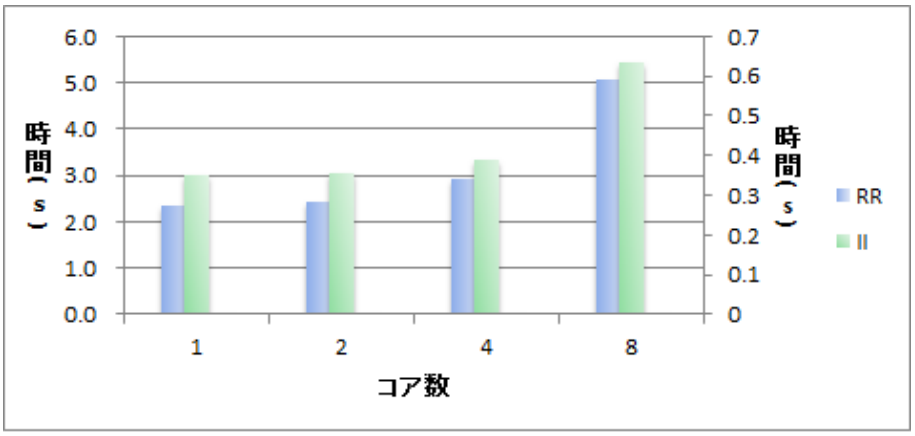

図 4.8: バス接続での予測平均実行時間 (RR:左軸, II: 右軸)

4.5.2 2次元メッシュネットワーク接続

図 (4.7) に示した各構成のネットワーク接続の予測平均実行時間を表 (4.5) に、またRRとIIのグラフを図 (4.9) 示す。

表 4.5: ネットワーク接続の予測平均実行時間 (s)

コア数	構成	RR	RI	II
1	MC	19.928	3.388	0.475
2	M2C	19.942	3.432	0.480
2	MCC	28.054	4.053	0.545
3	M3C	19.948	3.505	0.488
3	M2CC	25.365	3.908	0.530
3	MC2C	30.766	4.307	0.572
4	M4C	19.962	3.63	0.497
4	M3CC	24.015	3.904	0.529
4	M2C2C	28.072	4.187	0.560
5	M3C2C	26.458	4.207	0.559
8	M4C4C	28.105	5.227	0.657

また、RR, RI, II において 各コアの予測終了時間の表を表 (4.7), 表 (4.8) に示す。表より、メモリまでのホップ数が同じコアは実行時間にほとんど差が無いことが分かる。また、8コアの場合に RI, II のモデルではメモリから1ホップの場所にある4つのコアの予測終了時間は、バスに8コアを付けた場合の終了時間よりも短い事がわかる。

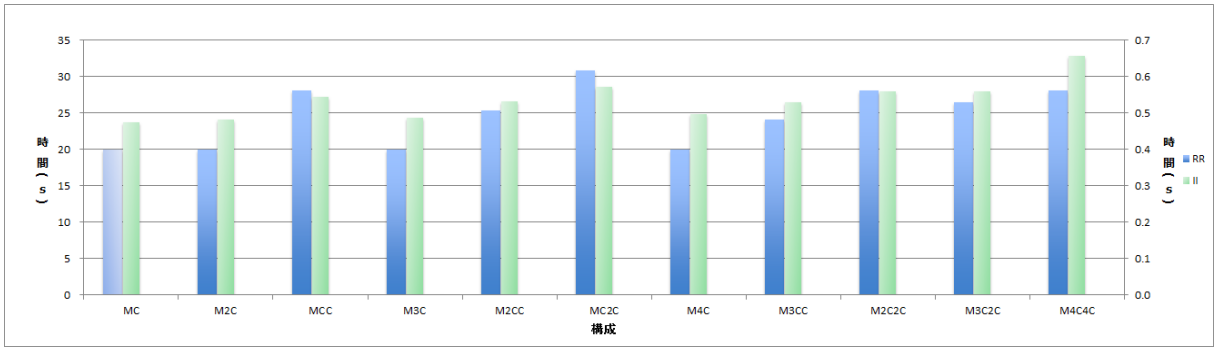

図 4.9: メッシュ接続での予測平均実行時間 (RR:左軸, II: 右軸)

表 4.6: 各コアの予測終了時間 (ネットワーク接続, RR)

コア数	構成	cpu0	cpu1	cpu2	cpu3	cpu4	cpu5	cpu6	cpu7	平均
1	MC	19.928								19.928
2	M2C	19.944	19.941							19.942
2	MCC	36.170	19.939							28.054
3	M3C	19.946	19.950	19.949						19.948
3	M2CC	36.190	19.952	19.954						25.365
3	MC2C	36.163	36.179	19.954						30.766
4	M4C	19.960	19.966	19.962	19.962					19.962
4	M3CC	36.189	19.958	19.955	19.957					24.015
4	M2C2C	36.183	36.194	19.955	19.958					28.072
5	M3C2C	36.205	36.204	19.961	19.961	19.958				26.458
8	M4C4C	36.246	36.225	36.216	36.240	19.978	19.975	19.981	19.980	28.105

表 4.7: 各コアの予測終了時間 (ネットワーク接続, RI)

コア数	構成	cpu0	cpu1	cpu2	cpu3	cpu4	cpu5	cpu6	cpu7	平均
1	MC	3.388								3.388
2	M2C	3.432	3.432							3.432
2	MCC	4.689	3.417							4.053
3	M3C	3.505	3.505	3.504						3.505
3	M2CC	4.763	3.481	3.480						3.908
3	MC2C	4.731	4.731	3.457						4.307
4	M4C	3.626	3.625	3.627	3.625					3.626
4	M3CC	4.872	3.581	3.580	3.582					3.904
4	M2C2C	4.829	4.830	3.544	3.543					4.187
5	M3C2C	4.982	4.983	3.690	3.689	3.689				4.207
8	M4C4C	5.863	5.856	5.859	5.860	4.595	4.595	4.596	4.591	5.227

表 4.8: 各コアの予測終了時間 (ネットワーク接続, II)

コア数	構成	cpu0	cpu1	cpu2	cpu3	cpu4	cpu5	cpu6	cpu7	平均
1	MC	0.475								0.475
2	M2C	0.480	0.480							0.480
2	MCC	0.610	0.479							0.545
3	M3C	0.488	0.488	0.487						0.488
3	M2CC	0.619	0.486	0.486						0.530
3	MC2C	0.616	0.616	0.485						0.572
4	M4C	0.497	0.497	0.497	0.497					0.497
4	M3CC	0.629	0.495	0.495	0.495					0.529
4	M2C2C	0.626	0.626	0.493	0.493					0.560
5	M3C2C	0.640	0.640	0.505	0.505	0.505				0.559
8	M4C4C	0.726	0.727	0.727	0.727	0.588	0.588	0.588	0.588	0.657

第5章

考察

本節では前節で得られたシミュレーション結果について考察を行う。

5.1 バス接続評価環境について

表 (4.4), 図 (4.8) I より RR (RI), の両方の環境において 8 コアを接続した時から性能が大きく低下することが分かる。4 コア接続時には RR 環境において約 1.48%, II 約 1.18 倍の実行時間増大にとどまっており、単純なバス接続は 4 コア程度が良いことが分かる。

5.2 メッシュ接続評価環境について

表 (4.5), 図 (4.9) より同じホップ数にあるコアの処理能力はほぼ同じであることが分かる。また、MC と M4C の予測平均実行時間を比べると RR で 1%以下、 RI で 7%程度、II においては 4%程度の実行時間増大にとどまっており、今回の評価環境においてはあるメモリから 1 ホップしか離れていないコア同士がは同時にそのメモリを使用しても性能にあまり影響が無いことが分かる。これは、コアの接続が増えたことによるメモリ要求の増大よりも、1 ホップにかかるレイテンシの効果が大きいためであると考えられる。

5.3 RR 評価環境について

表 (4.4) と表 (4.5) を比較すると、RR の時、メッシュ接続ではコア:メモリが一対一の関係にある時の実行時間は、バスでつないだ場合の 8.5 倍以上かかっている。コア 8 つをバス接続した場合と比べても 3.9 倍以上の実行時間である。これは、ルータ間通信のレイテンシ 320ns (100MHz の動作周波数で 32 サイクル) が、DRAM のアクセスレイテンシ 50ns よりも 6 倍以上遅く実行時間のほとんどが通信時間となっているためであると考えられる。明らかに、ルータ間

の通信速度が上がらなければ、現在のFPGA環境においてメッシュ接続ネットワークを採用したコンピュータシステムで性能を出すことは難しい。

5.4 RI, II 評価環境について

一方、RI, II の場合はコア:メモリが 1:1 の時の実行時間の増大は (RR バス、II バスと比べて) 約1.4倍程度となっている

しかし、8つのコアを接続する場合を考えるとバス接続でメッシュ接続の差は3%程度の実行時間増大にとどまっており、またメモリに近い4コアの予測実行時間はバス接続されたコアの91%程度になっている。

RI, II 環境においてコア数が増えるとバスネットワークの性能につか付いた要因は1ホップにかかる時間が短かく (25ns = 2.5サイクル), メモリのバンド幅を使っていたためであると考えられる。

5.5 提案モデルの性能

以上より、本提案モデルにおいては1つのメモリに接続するコアの数が少ない場合は バス接続にくらべて 性能が低下することが分かった。

シミュレータは8つのコアが同時に1つのメモリを使うところまでを検討した。コア数が増えるに連れて性能が向上していくことが実行結果より分かった。提案モデルにおいてはコアの数を数十程度つなげることも可能であり、その際は独立にプログラムが動作すると仮定すれば、バス接続よりも高い性能が出ると考えられる。

ただし、メモリアクセスレイテンシが遅い場合 (RR 環境) ではコア数が増えても性能がでないため、メモリのアクセスレイテンシを短くする必要があることが分かる。これは例えば1つのノードに複数のコアとメモリを載せるといった解決策が考えられる。

第6章

結論

6.1 まとめ

本論文ではFPGAの性能向上や利用領域が拡大していること、アクセラレータの利用によるシステムの性能向上を背景として各構成部品をメッシュネットワークで接続したコンピュータシステムを提案した。

提案システムは CPU や 特定アプリ向けのアクセラレータを FPGA に実装し、それらとメモリやI/Oポートをぞれぞれ1ノードとしてメッシュネットワークに接続する構想である。

提案システムの初期検討として簡略化したモデルのメモリ性能をシミュレータを用いて測定し、従来のバスネットワークの場合の性能と比較した。シミュレーション結果より現在のFPGA 性能ではルータ間の通信速度がボトルネックになってしまうためバスネットワークに比べて大きく性能が低下することがあることがわかった。将来的に周波数や、ルータ間の通信速度が向上した場合には性能低下を低く抑えてシステムを構成できる可能性がある。

6.2 今後の課題

今後の課題を以下に列挙する。

- より正確なシミュレーション環境を実現する
  今回はメモリのアクセスが1つのコアにおいて高々同時に1つしか発生しないと仮定したが、実際には複数のメモリ要求を同時に出し、各ルータがそれらを処理することでメッシュネットワークの性能を引き出すことができると考えられる。ただし、このシミュレーションを実現するためには既存のシミュレータに大幅に手を加える必要があると考えられる。

- 効率のよいスケジューリング機構の考案
  各サブコアのプログラムや使用するメモリ領域の割り当てを効率よく実

現するためのアルゴリズムが必要となる。これを実現するための第一歩ととして、サブコアを制御するためのデバイスドライバを開発中である

謝辞

研究を進めるにあたり、適切なご指導を賜りました指導教員の吉瀬謙二先生に深く感謝いたします。

また本研究で扱ったコンピュータシステムについて共同で研究している松田裕貴先輩、小川愛理さんとは幾度と研究に関する様々な議論を行いました。感謝いたします。

研究で疲れている時にカフェインを供給していただき、また NoC に関する点で多くの助言を頂いた森悠先輩に感謝いたします。

小林諒平先輩、Thiem Van Chu 先輩には本論文の執筆に関して、直前まで内容や構成について助言いただいたことに感謝いたします。

最後に吉瀬研究室のメンバーとして研究に限らず様々な事柄に助言をいただいた奥村開里先輩と臼井琢真君に感謝いたします。

参考文献

[1] L. Benini and D. Bertozzi. Network-on-chip architectures and design methods. *Computers and Digital Techniques, IEE Proceedings* -, Vol. 152, No. 2, pp. 261–272, Mar 2005.

[2] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, Rathijit Sen, Korey Sewell, Muhammad Shoaib, Nilay Vaish, Mark D. Hill, and David A. Wood. The gem5 simulator. *SIGARCH Comput. Archit. News*, Vol. 39, No. 2, pp. 1–7, August 2011.

[3] N.L. Binkert, R.G. Dreslinski, L.R. Hsu, K.T. Lim, A.G. Saidi, and S.K. Reinhardt. The m5 simulator: Modeling networked systems. *Micro, IEEE*, Vol. 26, No. 4, pp. 52–60, July 2006.

[4] S. Kumar, A. Jantsch, J.-P. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, and A. Hemani. A network on chip architecture and design methodology. In *VLSI, 2002. Proceedings. IEEE Computer Society Annual Symposium on*, pp. 105–112, 2002.

[5] Milo M. K. Martin, Daniel J. Sorin, Bradford M. Beckmann, Michael R. Marty, Min Xu, Alaa R. Alameldeen, Kevin E. Moore, Mark D. Hill, and David A. Wood. Multifacet's general execution-driven multiprocessor simulator (gems) toolset. *SIGARCH Comput. Archit. News*, Vol. 33, No. 4, pp. 92–99, November 2005.

[6] 松田裕貴・小川愛理・味曽野智礼（東工大）・藤枝直輝・市川周一（豊橋技科大）・吉瀬謙二（東工大）. Mierusys プロジェクト：複数の fpga を用いた先進的な計算機システムの開発. 2015.