3次元積層SiPを用いたマルチコアシステムのためのサイクルアキュレートシミュレータCubeSimの開発*

小島　拓也[†a)]　　池添　趄治[†]　　天野　英晴[†]

CubeSim: A Cycle Accurate Simulator for Multicore System with 3D SiP*

Takuya KOJIMA[†a)], Takeharu IKEZOE[†], and Hideharu AMANO[†]

あらまし　近年，IoTデバイスや組み込み機器で要求される機能，性能，インターフェースが多様化している．これを，単一のSoC (System On a Chip) で解決するのはコストの観点から現実的ではない．そこで，複数のLSIチップを3次元積層技術で組み合わせるSiP (System In a Package) が注目されている．特に，TCI (Thru-Chip Interface) と呼ばれるチップ間無線通信技術による積層システムは用途に応じて多様なシステムを構築できる．このようなシステムの評価を行う場合ハードウェア記述言語を用いたRTLシミュレーションでは所要時間が長く，積層されるアクセラレータやシステム全体の設計を最適化するのが困難であった．また，時間のかかるRTLシミュレーションではアプリケーション開発におけるデバッグが非効率化してしまう．そこで，本研究では積層システム向けのサイクルアキュレートシミュレータCubeSimを開発した．プロセッサのキャッシュサイズなどをパラメータ化したことによって，システムの設計探索を可能にした．CubeSimは積層アクセラレータ用の抽象クラスを提供し，動作をシミュレートする最低限のコードで，システムの評価を行うことができる．評価の結果，CubeSimは商用のRTLシミュレータと比較して最大で234倍高速にシミュレーションを行うことが可能となった．更に，実チップ化されている2種類のアクセラレータをCubeSimに組み込み，システム全体の評価及び設計探索を行った．システム構成の変更により最大で約38%のエネルギー削減が期待できると明らかにした．

キーワード　3次元積層LSI，SiP，TCI，サイクルアキュレートシミュレータ

1. まえがき

近年，組み込み機器やIoTデバイスで要求される性能や機能が多様化している．したがって，用途ごとに単一のSoC (System On a Chip) を設計，製造するのはNRE (Non-Recurring Engineering) コストの高さゆえ現実的な手段ではない．そこで，プロセッサコアやメモリ，IOインターフェースなどのモジュール化された比較的小さなLSIチップを製造し，それらを組み合わせたSiP (System In a Package) によるソリューションが注目されている．用途に応じて利用するモジュールを選択すれば良いため，短期間でシステムの設計ができる．機能モジュールの製造面ではチップのダイサイズが小さくなることにより，チップダイ単体の歩留まり向上が期待できる．

SiP技術は大きく分けて，シリコンインターポーザも用いた2.5次元実装と，複数枚のチップを垂直方向へ積層した3次元実装が存在する．いずれの技術もTSV (Through Silicon Via) などのチップ間通信技術が用いられる．2.5次元実装では複数のダイを平面状に配置するためスケーラビリティに限界があるが，3次元実装ではチップ同士はより高密度に接続され高い転送バンド幅を得る．3次元実装ではシステムを小さなパッケージへ高密度に集積できるため，配線長の削減などによる高性能化も得られる．しかし，TSVは特殊な製造工程を必要とし，チップ単体の製造コストやデザインルールが増加してしまう．そのため，このようなコスト増加に見合うキラーアプリケーションは限定的であり，市場では3次元積層技術はHMC [1] やHBM [2] などのメモリチップに限られていた[3]．しかし，近年はTSV製造技術やパッケージング技術の

† 慶應義塾大学大学院理工学研究科，横浜市
Graduate School of Science and Technology, Keio University, Hiyoshi 3–14–1, Kohoku-ku, Yokohama-shi, 223–8522 Japan
a) E-mail: wasmii@am.ics.keio.ac.jp
* 本論文は，システム開発論文である．
DOI:10.14923/transinfj.2020PDP0046

向上により SoIC [4] や Foveros [5] などロジックチップ同士の積層も利用可能になりつつある．

対して，3 次元積層のための技術としてチップ間無線通信用インターフェース TCI (Thru-Chip Interface) が提案されている [6]．TCI ではコイルの磁束変化を用いて通信を行うため，TSV のような物理的接続が不要であり，通信用のコイルは CMOS プロセスの配線層を用いるため設計に特殊な EDA のサポートや，専用の製造プロセスを必要としない．TCI を利用したシステムのプロトタイプとして汎用のプロセッサやアクセラレータを積層した Cube-1 [7] や Cube-2 [8] が報告されている．

しかしながら，TCI を用いて複数のチップを積層するヘテロジニアスなマルチコアシステムを設計し，利用する場合に次のような課題が存在する：(1) シミュレーション，デバッグ環境が乏しい，(2) システム全体の設計探索が困難である (3) アクセラレータの新規開発のための予備評価ができない．現在，アプリケーションの動作検証には設計資産の RTL を用いたシミュレーションを行う．したがって，実用的で複雑なアプリケーションを想定した場合，シミュレーションにかかる時間とデバッグの困難さが問題となる．また，設計済みの RTL を用いたシミュレーションを利用する場合，メモリアクセス性能や CPU のキャッシュサイズ，アクセラレータのロジック規模などのアーキテクチャレベルのパラメータを変更して設計探索を行うのが困難となる．更に，本論文が想定する積層システムに異なるアプリケーション領域のためのアクセラレータを新規開発する場合，ハードウェア記述言語で一から設計しなければシステム全体の評価を行うことができない．評価の結果，高速化，高効率化が期待できないとわかれば，仕様を検討し直し設計し直さなくてはならない．

本研究ではこれらの問題を解決するために，下記の特徴をもつ TCI を用いた 3 次元システム向けのサイクルアキュレートシミュレータ CubeSim を開発する．

- サイクルアキュレートな高速シミュレーション
- GDB による効率的なデバッグ環境の提供
- キャッシュやバス構成のパラメータ化
- 積層アクセラレータ用の抽象クラスの提供

本論文の構成は以下のとおりである．**2.** では既存のシミュレータを紹介し，本研究で開発したシミュレータ CubeSim の独自性に関して言及する．次に，**3.** において CubeSim が対象とする 3 次元積層システムの概要を説明する．**4.** では，開発した CubeSim の実装方法を述べる．そして，**5.** では CubeSim の評価，及び CubeSim を用いた設計探索の利用例を示す．最後に，**6.** で本論文の結論を述べる．

2. 関連研究

プロセッサシミュレータはシステムの設計者にとって必須のツールであり，これまでに多数のシミュレータが提案されている．これらのシミュレータはサポートされる ISA，シミュレーションの精度などで分類できる．x86 は汎用の PC やサーバなどで広く利用されるため，そのシミュレータの数も多い [9]．しかし，本研究が想定する組み込み機器などのデバイスにおいては積層されるプロセッサに x86 のような複雑で大規模な ISA を採用する可能性は低い．よって，本章では本研究が想定する MIPS をサポートするシミュレータに焦点を当てる．

gem5 [10] や QEMU [11] などのシミュレータは MIPS を含む多くの ISA をサポートするマルチターゲットのシミュレータであるが，拡張性の高さ故にソースコード量は膨大であり，拡張のための人的コストが高い．本研究で開発した CubeSim は，アクセラレータの設計者が容易にアイディアを形にできることを目指している．よって，ソースコードの簡潔さを意識した MIPS のシングルターゲットシミュレータとしている．

また，シミュレータは機能レベルでのシミュレーションが可能ないわゆるエミュレータと，タイミングレベルの精度でシミュレーションを行うもの，またはその両方が可能なシミュレータに大別される．前者はソフトウェアの動作検証などに用いることができ，一般に高速に動作するが詳細な実行時間の見積もりはできない．MIPS 向けとしては SPIM [12] がこれに分類される．gem5 はサイクルアキュレートなシミュレータであるが，前者に該当する CPU モデルも含まれる．SIMICS [13] も同様に高速にエミュレーションを行うモードと，正確にタイミングシミュレーションを行うモードの両方を備える．

CubeSim は **4.1** で説明するように VMIPS [14] を拡張して開発している．VMIPS も元々は前者に分類されるエミュレータであるが，可読性が高く，メモリマップト I/O デバイスなどの追加，拡張が容易なモジュールベースの設計となっている．**3.** で説明するように積層されるアクセラレータチップはメモリ空間にマップさ

れるため，VMIPS は CubeSim を開発するにあたり都合の良いものである．本研究ではこれを **4.** で説明する改良によって，サイクルアキュレートなシミュレータへ拡張している．したがって，CubeSim は軽量さを維持しつつ gem5 と同じく任意のアクセラレータを追加可能なシミュレータを目指しており，Multi2Sim [15] における GPU のような特定のアクセラレータをのみをサポートするものではない．また，CubeSim が対象とするシステムは **3. 2** で説明するように積層チップ同士がパケット方式の通信を行う．したがって，CubeSim は対象システムで利用される通信プロトコルの相互接続ネットワークもシミュレーションする．gem5 にも Garnet [16] と呼ばれるオンチップネットワークモデルが備わっており，CubeSim と同様にクレジットベースの通信をシミュレーションする．しかし，Garnet はオンチップネットワークを念頭に置いており，転送先ルータのバッファ状態を専用のクレジットリンクで通知するモデルであるのに対し，CubeSim ではデータとクレジット情報が一つの TCI で共有されるチップ間ネットワークを想定している点が異なる．

3. シミュレーション対象のシステム

本論文が提案する CubeSim はビルディング型計算システムと呼ばれる複数の LSI チップが TCI によって相互接続されたシステムをシミュレーションの対象とする．TSV (Through-Silicon-Via) のような物理的接続を必要としないため，チップ製造後に用途に応じて複数のチップを選択し，柔軟に組み合わせることができるシステムである．既に Cube-1 [7] と Cube-2 [8] がプロトタイプとして開発され，評価及び動作検証が行われている．Cube-1 は MIPS R3000 互換の汎用 CPU Geyser [17] と低消費電力 CGRA (Coarse-Grained Reconfigurable Architecture) の CMA (Cool Mega Array) [18] を 3 次元的に積層したシステムとして報告された．Cube-2 では TCI による通信機構を IP 化し [19]，CMA 以外に CNN (Convolutional Neural Network) アクセラレータ SNACC [20] と Non-SQL データベースアクセラレータ KVS [21] が利用可能なアクセラレータとして追加された．

3.1 TCI によるチップ間通信

誘導結合を利用した通信を行うために，TCI ではメタル層に正方形のコイルを形成する．一つのコイルは送信用または受信用として利用される．送信及び受信チャネルを形成するために，各チャネルは同期用クロックとデータ用の二つのコイルをもつ．Cube-1 では積層チップ間で単方向リングネットワークを形成していたが，Cube-2 では図 1 に示すように，上方向送信チャネル (Tx)，受信チャネル (Rx) 及び下方向 Tx，Rx の計 4 チャネルを用いたエスカレータネットワーク [22] を形成する．つまり，エスカレータネットワークを形成するのに 1 チップあたり計 8 個のコイルが必要となる．

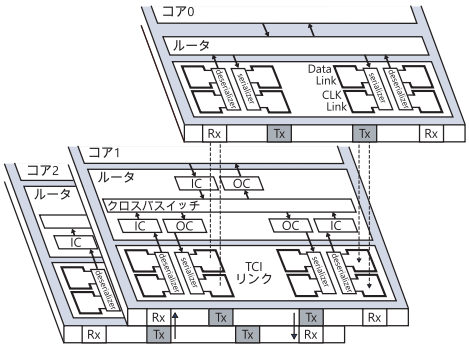


図 1　TCI-IP によるエスカレータネットワーク

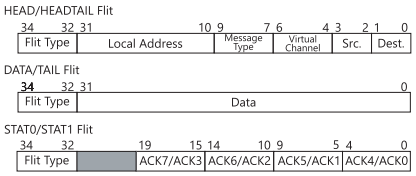


図 2　Cube システムにおけるパケットフォーマット

3.2 ルータによるフロー制御

TCI-IP には SERDES 及び 2.5GHz の内部発振回路が含まれ，最大 50MHz で動作する 35bit 幅のチップ間データ転送チャネルとして設計されている．エスカレータネットワークでは図 2 に示す 35bit を 1 フリットとする可変長パケットが TCI-IP を経由して他のチップへ転送される．データの読み出し及び書き込みにはシングル転送モード (1word/4Byte 転送) とブロック転送モード (16word/64Byte 転送) が用意されている．シングル転送モードでは HEAD フリット +TAIL フリットの 2 フリットで一つのパケットを構成し，ブロック転送モードでは HEAD フリット +15 個の DATA フリット +TAIL フリットの計 17 フリットで一つのパケットを構成する．TCI-IP を搭載したチップは図 1 に示すように三つの入力チャネル (IC) と三つの出力チャネル (OC) をもつルータが備わっている．三つのチャネルはそれぞれチップ内のコア及び上層チップ，下層

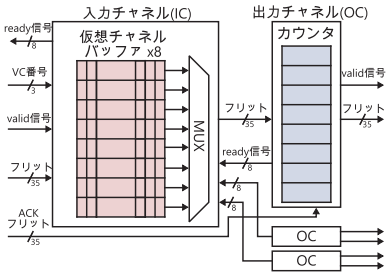


図 3　ルータの入出力チャネル

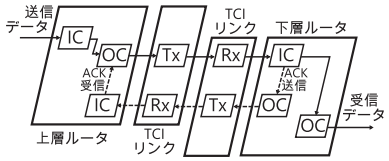


図 4　チップ間のデータ転送の様子

チップとのデータ転送に利用される．各チップには最上層のチップから順に ID が振られ，ルータはヘッダフリットに含まれる宛先 ID (Dest.) を元に経路を決定する．現在の設計ではヘッダフリットのチップ ID 用フィールドは 2bit であり，最大 4 枚のチップを積層可能である．

図 3 にルータの入出力チャネルの概要を示す．一つの入出力チャネルは物理的には一つのリンクであるが，仮想的に八つのチャネル (VC0〜VC7) に分割されている．入力チャネルには各仮想チャネルごとにバッファが存在し，経路が決定したパケットのうちもっとも高い優先度をもつ仮想チャネルが出力チャネルにデータを転送する．現在の設計では VC0 がもっとも高い優先度をもち，VC0 から VC7 の順に優先度が低くなる．

TCI は無線接続であるため，転送先の入力チャネルバッファに空きがあるかどうかをピギーバックによって確認している．図 4 にはある上層チップからすぐ下の下層チップへデータを転送し，ACK を受け取る様子が示されている．実線で表される流れが転送データで，破線で表される流れがピギーバックのための ACK である．ピギーバックには図 2 に示した STAT0 または STAT1 フリットを用いている．この例では，上層の出力チャネルがパケットを送信すると，そのフリット数だけカウンタを増やす．下層の入力チャネルは届いたパケットをバッファし，次の転送先へデータの転送が完了すると，出力チャネルに ACK の送信を行う

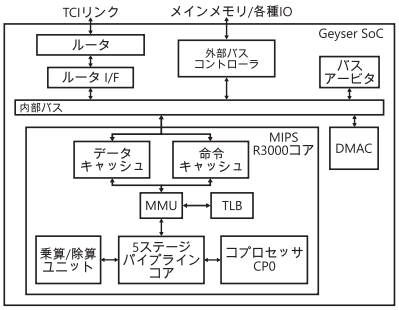


図 5　Geyser の構成

よう指示を出す．例えば，このデータ転送に VC0 を利用していたとすると出力チャネルは STAT0 フリットの ACK0 フィールドに転送したフリット数をセットし，上層へ送り返す．上層の出力チャネルは STAT0 フリットが届くと，フィールドにセットされた数の分だけカウンタをディクリメントする．したがって，出力チャネルはカウンタの値によって転送先入力チャネルに空きがないと判断すると，ready 信号を下げる．ただし，チップ内のローカルコアへデータを転送する場合は信号線によるハンドシェイクが可能であるためこれは不要である．

3.3　ホストプロセッサ Geyser

本研究では Cube-1 及び Cube-2 と同様にホスト CPU である Geyser を利用するシステムを想定する．Geyser は MIPS R3000 命令セットとの互換がある CPU であり，図 5 に示す典型的な 5 段パイプラインのコアと各々 4KB の 2-way セットアソシアティブな命令キャッシュとデータキャッシュ，システム制御用コプロセッサ CP0，乗除算ユニットなどから構成される．CP0 や TLB によって仮想メモリのサポートや例外処理などを行うことが可能で，Linux OS や組込向け OS の TOPPERS などの移植が報告されている [24]．MIPS R3000 のコアは DMAC やルータとともに内部バスで結ばれている．MIPS R3000 の命令セットに加えて，Geyser には MIPS32 互換のキャッシュ命令を備えており，明示的にキャッシュを操作することができる．

MIPS R3000 は 32bit の仮想アドレス空間を 2GB の kuseg，0.5GB の kseg0 と kseg1，1GB の kseg2 の四つのセグメントに分割する．このうち，ルータ及び積層されたチップは kseg0 及び kseg1 の一部にマップされている．それぞれのセグメントで積層チップ用に 16MB の領域が確保されており，CPU コアがこれらの領域へ読み出し要求や書き込み要求を出すと，それ

がルータインタフェースへ伝わり，パケットに変換される．ルータインタフェースは 16MB 内のオフセット (24bit) のうち上位 2bit で，宛先のチップ ID を決定する．読み出し要求の場合，ルータインタフェースに読み出しデータが到着するまで CPU コアはストールする．一方で，書き込み要求の場合はパケットが送信され次第ストールが解消する．kseg0，kseg1 ともに TLB を用いたアドレス変換が無効の領域となっており，カーネルモードでのみアクセス可能な領域となっている．kseg0 へのアクセスはキャッシュが利用されるのに対し，kseg1 へはそのままアクセスされる．Cube システムでは kseg0 のアドレスで積層チップにアクセスするとキャッシュミスが発生し，ルータはブロック転送モード (17 フリット) でデータをフェッチする．したがって，キャッシュブロックサイズはブロック転送モードのパケットデータサイズと同じ 64 Byte となっている．一方，kseg1 のアドレスで積層チップにアクセスするとそのままシングル転送モードでデータが読み書きされ，キャッシュされない．

4. サイクルアキュレートシミュレータ CubeSim の設計と実装

4.1 設計方針

あるシステムでアプリケーションの実行時間を見積もるためだけに RTL シミュレーションを行うのは過剰であり，また，システムの規模が大きくなるにつれて所要時間も大きくなる．更に，キャッシュやバス幅などのシステム構成をパラメータ化して，設計探索を行うのは容易ではない．アプリケーション自体のデバッグ環境が乏しいという問題もある．そこで，本研究では **3.2** で説明したパケット交換方式を用いた 3 次元積層システムのサイクルアキュレートシミュレータ CubeSim を開発し，評価を行う．

2. で述べたとおり，CubeSim はオープンソースの MIPS R3000 シミュレータ VMIPS [14] をベースにしている．C++ 言語で記述されており，必要に応じてメモリマップトデバイスの追加や，ハードウェア割り込みなどが容易にカスタム可能である．また，GDB によるリモートデバッグをサポートする．VMIPS では命令セット実行パートにおけるサイクル数に関しては遅延スロットも含め正確にエミュレートされているものの，メモリアクセスは理想化されており，全てが 1 サイクルで完了する想定となっている．そのため，キャッシュのシミュレーションは事実上存在しない．また，正しくパイプラインをシミュレーションしているわけではないため，各種パイプラインハザードが考慮されていない．そこで，本研究では VMIPS をベースに下記の拡張を行う．

- 5 段パイプラインの正確なシミュレーション
- キャッシュ命令のサポート
- メモリアクセスレイテンシのパラメータ化
- n-way セットアソシアティブキャッシュ
- バスアービタ及び DMAC の追加
- ルータ及び周辺回路の追加
- 積層アクセラレータ用の抽象クラス
- 積層アクセラータ用のデバッガ

4.2 パイプラインシミュレーション

VMIPS ではキャッシュミスなどによるストールが起きない想定となっている．一方で，Cube システムのような積層システムでは全体の処理に対してデータ転送にかかる時間が小さくない．したがって，CPU コアのストールを正しくシミュレーションする必要がある．ストールの発生要因はステージごとに異なるため，固定のレイテンシであると想定するのは適切ではない．表 1 に Geyser におけるストール発生要因と発生するステージを示す．MIPS R3000 では遅延分岐を採用しているため制御ハザードは起きない．Geyser ではデータハザードの多くはフォワーディングにより解決している．しかし，分岐評価は ID ステージで行われるため，二つ先の MEM ステージから供給されるロード結果を分岐評価に利用する場合，1 サイクルストールする．また，動作周波数の低下を防ぐために，乗算と除算，またコプロセッサ命令は各々 3 サイクル，9 サイクル，5 サイクルのマルチサイクル実行となっており，構造ハザードによるストールが発生し得る．キャッシュミスを含むメモリアクセスに起因するストールは読み出しの場合はデータが届き次第，または書き込みの場合データを送信し次第解消するため，固定のサイクル数ではない．特に，ルータ経由で積層チップへアクセスする場合，トラヒックの混雑状況にも依存する．

表 1　Geyser におけるストール発生要因

ステージ	要因	ストールサイクル
IF ステージ	メモリアクセス (命令)	不定
ID ステージ	データハザード	1 サイクル
EX ステージ	構造ハザード (CP0)	最大 5 サイクル
	構造ハザード (積算器)	最大 3 サイクル
	構造ハザード (乗算器)	最大 9 サイクル
MEM ステージ	メモリアクセス (データ)	不定

CubeSim は C++ 言語で記述されたプログラムとして実行されるため，当然シーケンシャルに実行される．そのため，パイプラインステージを上から順に実行すると，後方のステージで発生したストールに対処できない場合がある．そこで，CubeSim では各ステージをハードウェアの状態を変更しないパートと変更するパートに分離し，前者でストール発生の有無を確認する．全てのステージでストールが発生しない場合，後者のパートを実行する．また，例外の取り扱いに関しても同様の問題が発生する．例えば，前方のステージで発生した例外 (e.g.，アドレスエラー例外や予約命令例外) を発生した直後にハンドルしてしまうと，本来実行されるはずであった先行するステージが実行されなくなってしまう．または，先行するステージがそのすぐ後に例外を起こす場合，それを先に処理しなくてならない．そこで，CubeSim では発生した例外はいったんペンディングされ，MEM ステージのメモリストアまたは，WB ステージのレジスタ書き込みが行われる前にペンディングされた例外がハンドルされる．このように，CubeSim では Geyser における各パイプラインステージの動作を正確にシミュレーションすることができ，**5.1** で評価するようにこのパイプライン部における実行サイクル数は実際の RTL 設計と完全に一致する．

4.3 キャッシュ命令のサポート

MIPS R3000 には明示的にキャッシュを操作する命令は備わっていないが，Geyser には MIPS32 命令セットなど後継の命令セットに含まれる *cache* 命令をサポートしている．前述のとおり積層チップ間はワイヤレスに接続され，パケット通信によりデータ転送を行うため，スヌープキャッシュプロトコルなどを用いてキャッシュコヒーレンシを保つのが難しい．アクセラレータチップがマップされるアドレス空間において当該アクセラレータがデータを書き換える可能性がある場合，プロセッサ側でキャッシュブロックを強制的に invalidate するような機能は重要である．したがって，CubeSim においても *cache* 命令をシミュレートできるよう拡張を行っている．

4.4 アーキテクチャのパラメータ化

前述のとおり，CubeSim ではメモリアクセスをより実システムに近い形でシミュレートできる．一般に，メモリアクセス要求を出してからデータを受け取るまでに複数サイクルを必要とする場合が多い．CubeSim ではこの所要サイクル数をパラメータ化し，様々な設計に対応できるようになっている．また，外部のメインメモリを想定する場合，そこでも更なる遅延時間が発生し得る．CubeSim ではメモリ空間にマップされるデバイスごとに異なる遅延時間を設定できるようになっている．ただし，本遅延モデルではアクセスパターンによらず固定されたサイクル数のレイテンシとなる．したがって，シミュレーション対象のシステムがアクセスパターンによって異なるレイテンシをとる設計の場合，シミュレーション結果に誤差が生じる．この影響に関しては **5.1** で議論する．

現在の Geyser の設計では命令キャッシュ，データキャッシュともに 64B キャッシュラインを 64 ブロック × 2way 計 8KB の容量をもつ．しかし，CubeSim ではこれらの構成は全てパラメータ化されており，対象システムに最適なキャッシュ構成を探索することが可能である．同様に内部バスのデータ幅もパラメータ化されており，例えばキャッシュがメインメモリからデータをフェッチする際に 1 サイクルに複数ワードのデータを受け取ることが可能な構成をシミュレーションすることができる．

4.5 バスアービタと DMAC

VMIPS では内部バスにつながるデバイスは MIPS R3000 コアのみであり，そのほかのバスマスタを想定していない．そこで，バスアービタを追加し，内部バスにつながるモジュールは排他的にバスを利用するようにシミュレートされる．例えば，本研究で想定する Geyser では MIPS R3000 コアと DMAC がバスマスタとなり，排他的にバスを利用する．バスマスタの数に制限はなく，所定の抽象クラスを継承したデバイスインスタンスをバスマスタとしてバスに接続するだけで，シミュレーションが可能になる．

4.6 ルータ及び積層されるチップ

CubeSim には **3.2** で述べたチップ間のデータ転送を行うルータや Geyser におけるルータインターフェースも含まれる．ルータの内部はパイプライン化されており，次のルータへデータが転送されるには最短でも 4 サイクル所要する．ルータによるデータ転送はこのようなパイプライン処理や，仮想チャネルなども含め正確にシミュレートされる．また，仮想チャネルのバッファサイズもパラメータ化されており，バッファサイズ変更の影響を評価することができる．

積層されるアクセラレータコアを容易に追加できるように，ルータとのインターフェースが実装済みの抽象クラスを提供する．したがって，システム設計者は

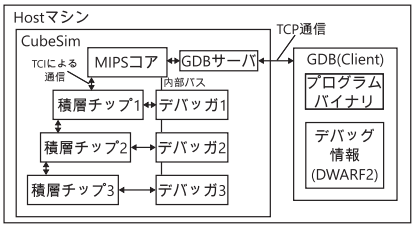


図 6　GDB によるデバッグ環境

このクラスを継承し，アクセラレータコアの動作部を記述しオンチップメモリなどの構成などを指定するだけで，新たなアクセラレータをプロセッサと積層してシミュレーションすることが可能となる．

4.7　積層アクセラータ用のデバッガ

VMIPS では標準的に利用されているデバッガ GDB が利用できる．CubeSim においてもこれを利用できるように，上述の拡張を行っている．図 6 に示すように，CubeSim をデバッグモードで起動すると GDB サーバが合わせて起動し，GDB はこのサーバへリモートターゲットとして接続する．MIPS 用 GDB ではメモリの内容やレジスタファイル，プログラムカウンタの値を読み書きでき，ブレークポイントの設定も可能である．CubeSim ではデバッグモード時に図 6 に示すデバッグ用の専用モジュールを Geyser のメモリ空間にマップし，仮想的に内部バスへ接続する．これによって，GDB から通常のメモリアクセスとしてアクセラレータ内部の状態を確認することができる．ただし，アクセラレータ側は所定のメンバ関数をオーバーライドすることによって所望のモジュールのデータを読み書きできるようにしておく必要がある．また，アクセラレータの状態に応じてシミュレーションを停止させる機能をハードウェアブレークポイントとして実装することができる．

4.8　ソースコードの変更量

ここまで説明した改良を実現するにあたり，VMIPS の主要ファイルへの変更と，幾つかのファイルを追加した．変更したコード量 (削除した行と追加した行の合計) を表 2 にまとめる．ただし，クラス定義などのヘッダファイルは除き，実装部の変更のみを示した．cache.cc 以下は今回新たに追加したファイルである．パイプライン化やメモリアクセスなどによるストールのエミュレーションをするように改良したため，特に cpu.cc で大幅な変更を行っている．加えて，メモリアクセスのレイテンシを再現するために，バスにあたる mapper.cc も大きく変更されている．新規に追加したファイルはいずれも 1000 行を超えるような大きなファイルではなく，十分な軽量さ，可読性を維持している．

表 2　シミュレータの主要ファイルのコード変更量

ファイル名	内容	変更行数 (割合 %)
vmips.cc	TOP モジュール	322 (36%)
cpu.cc	CPU コア	3508 (135%)
cpzero.cc	CP0	28 (5%)
mapper.cc	内部バス	472 (80%)
debug.cc	GDB サーバ	79 (10%)
cache.cc	キャッシュ	547
busarbiter.cc	バスアービタ	30
router.cc	ルータ	688
routerinterface.cc	インターフェース	486
accelerator.cc	アクセラレータ抽象クラス	364

5.　評　　価

5.1　シミュレーションの精度と速度

本節では初めに本研究で開発した CubeSim が同等の RTL シミュレーション，及び広く利用されるサイクルアキュレートシミュレータ gem5 [10] と比べてどの程度の高速化を達成するかを評価する．RTL シミュレーションでは Verilog HDL で記述された Geyser の設計に対して Cadence 社 NC-Verilog (INCISIVE 15.20.020) を利用する．gem5 を用いたシミュレーションでは CPU モデルにインオーダー実行の TimingSimpleCPU，メモリモデルに SimpleMemory を SE (System call Emulation) モードで利用する．CubeSim のコンパイルには GCC 4.8.5 を用いるが，Clang での動作も確認されている．CubeSim，gem5 でのシミュレーションにおいてキャッシュの構成は **3.3** で述べた Geyser の設計と同一の構成とする．いずれのシミュレーションも Intel Core i5-4250U 1.30GHz，8GB DDR3-SDRAM のマシン環境で測定を行う．gem5 との比較を行うために，本節での評価に用いるアプリケーションは CPU 部のみで実行可能なものとする．また，Geyser 向けのライブラリ群はモデルの違いから gem5 で直接利用できない．そのため，特定のライブラリを必要としない CHStone [25] をベンチマークとして利用する．各種アプリケーションをシミュレーションしたサイクル数及びシミュレーションに要した時間を表 3 にまとめる．RTL シミュレーション及び CubeSim の実行結果については MIPS コアが実行サイクル数のうち動作中 (busy) とストール中 (stall) であったサイクル数の内訳

表 3　各種シミュレーションとの精度と速度の比較

App.	Geyser RTL シミュレーション		CubeSim			gem5		
	サイクル数	実行時間 (秒)	サイクル数	実行時間 (秒)	誤差	サイクル数	実行時間 (秒)	誤差
adpcm	157133 busy: 95596 stall: 61537	12.627	157495 busy: 95596 stall: 61899	0.356	0.230%	145819	0.639	7.18%
aes	56752 busy: 36620 stall: 20132	4.994	56239 busy: 36620 stall: 19619	0.196	0.903%	60541	0.57	6.74%
blowfish	947060 busy: 748518 stall: 198542	62.117	944498 busy: 748518 stall: 195980	0.431	0.271%	1082647	1.195	14.4%
gsm	31221 busy: 17614 stall: 13607	3.596	31303 busy: 17614 stall: 13689	0.051	0.263%	31522	0.543	0.960%
jpeg	4372209 busy: 2654450 stall: 1717759	235.649	4339199 busy: 2654450 stall: 1684749	1.804	0.755%	4139368	3.219	5.37%
mpeg2	24921 busy: 11263 stall: 13658	3.384	24827 busy: 11263 stall: 13564	0.065	0.377%	32356	0.558	30.0%
sha	756651 busy: 674022 stall: 82629	51.793	755738 busy: 67402 stall: 81716	0.392	0.121%	814455	1.089	7.65%

も示している．

今回評価に用いた Geyser の RTL 設計はメモリアクセスの際に，アクセス先モジュールやアクセス種 (読み出し，または，書き込み) によって MMU やバスでのレイテンシが異なる設計となっている．一方，CubeSim では **4.4** で述べたとおり一定のアクセス遅延でタイミングシミュレーションが行われる．したがって，シミュレーションされるサイクル数は完全には一致しない．現在利用可能な Geyser の RTL 設計は試作機向けのものであり，利用可能な I/O ピンが制限されていることと，保守的な設計であることからメモリアクセスのレイテンシが大きい．これに最も近くなるように，CubeSim における内部バスや MMU におけるレイテンシは 16 サイクル，外部メモリにおけるレイテンシを 48 サイクル (ブロックリード時) に設定した．その結果，誤差は 1% 以下であり，MIPS コアのパイプラインが動作する実行時間は完全に一致している．RTL シミュレーションと比較して，最大で blowfish で約 144 倍，最低でも aes で約 25 倍高速にシミュレーションが可能である．

gem5 のシミュレーションでは既存の CPU モデルを利用しているためパイプラインのモデルが Geyser の実設計及び CubeSim とは異なる．また，メモリモデルもやや異なるため最大で 30% 程度の誤差を含んでいる．しかし，この誤差を考慮しても CubeSim は gem5 と比較して十分高速にシミュレーションを行うことができている．

5.2　積層システムのシミュレーション

次に，本シミュレータが提供するアクセラレータ用の抽象クラスを用いて Cube-2 のファミリーチップである CGRA チップ CC-SOTB2 [26] 及び CNN アクセラレータ SNACC [20] を CubeSim へ組み込む．CC-SOTB2 は CMA の 1 種であり主に画像処理用のアクセラレータとして開発され，演算処理ユニットである PE (Processing Element) が計 96 個アレーに配置されている．タスクに応じて PE アレーの構成を変更し，効率的に計算を行う再構成可能デバイスの一種である．CC-SOTB2 を積層したシステムの評価には MiBench [27] から抽出した静止画像の JPEG エンコードのプログラムを用いた．一方，SNACC は CNN アクセラレータであり，チップ内に四つのコアをもつ．畳み込み演算に特化した SIMD 演算器，重みデータや入力データなどを保持するデータメモリをコアごとに独立してもっている．評価には [8] でも用いられた AlexNet [28] の FC7 層を選び，全てのコアを利用するようにプログラムされている．

JPEG エンコードに関しては 1) Geyser 単体，2) Geyser+CC-SOTB2，3) Geyser+CC-SOTB2x2 の 3 種類の構成を検討する．AlexNet に関しては 1) Geyser 単体，2) Geyser+SNACC の 2 種類の構成を検討する．

表 4　積層シミュレーション時間の比較 (秒)

	構成	NC-Verilog	CubeSim
JPEG エンコード	Geyser 単体	27.024	0.115
	Geyser+CC-SOTB2	24.750	0.171
	Geyser+CC-SOTB2x2	29.231	0.174
AlextNet FC7 層	Geyser 単体	4319.409	86.086
	Geyser+SNACC	2908.185	40.954

SNACC を複数チップ利用したシステムは後述する性能評価の結果から，更なるアプリケーションの高速化が期待できなかったため除外した．表 4 に CubeSim と NC-Verilog による RTL シミュレーションの所要時間をまとめる．単体 CPU のシミュレーション結果と同様に，サイクル数に若干の誤差を含むが，3 枚のチップが積層された Geyser+CC-SOTB2x2 の構成でも誤差は 0.6% ほどである．

積層チップを増やすと RTL シミュレーションでは回路規模が増大するため，それに伴いシミュレーション時間も増大すると考えられる．**5.1** における Geyser 単体でのシミュレーション結果では NC-Verilog は最も遅い場合 (mpeg2) で 1 秒間に約 7400 サイクルのシミュレーションを行っていた．対して，Geyser+CC-SOTB2x2 の構成では NC-Verilog は 1 秒間に約 7800 サイクルのシミュレーションを行っている．したがって，特に NC-Verilog では回路規模よりも，実際に動作するモジュールの割合が実行時間に大きく影響していると推測される．

一方で，CubeSim の場合は積層チップを増加させた分だけサイクルあたりのシミュレーション時間は増える．CubeSim 実行時間のプロファイルを行った結果，特にルータ部のシミュレーションに時間がかかっており，CC-SOTB2 と SNACC を用いた場合の両方で約 60% の時間を占める．次いで CC-SOTB2 や SNACC 部のシミュレーションに 30% ほどの時間を消費し，CPU 部のシミュレーション時間は全体の 10% 以下と非常に小さい．したがって，JPEG エンコードを用いた場合に Geyser 単体のシミュレーションで最大の 234 倍の高速化を達成しているものの，Geyser+CC-SOTB2 の構成のシミュレーションでは約 144 倍の高速化にとどまっている．また，Geyser 単体で AlexNet を動作させた場合の高速化は 50 倍程度にとどまる．これは，キャッシュミス率の違いが影響を与えている．JPEG エンコード実行時と比較して，AlexNet 実行時のキャッシュミス率は 10 倍近く，その結果キャッシュの動作をシミュレートするパートの実行頻度が増えたため CubeSim ではシミュレーションに時間がかかっている．

NC-Verilog が約 1GB のメモリ使用量なのに対し，CubeSim は約 10 分の 1 の 100MB 程度で実行可能である．よって，CubeSim はラップトップクラスの CPU を内蔵したホストマシンでも十分高速に積層システムのシミュレーションが可能である．

5.3　CubeSim を用いたシステムの設計探索

次に，前節と同じ二つのアプリケーションを対象として様々なシステム構成を CubeSim でシミュレーションし，性能とエネルギーの比較を行う．本評価では内部バスや MMU におけるレイテンシは合わせて 8 サイクルに設定している．また，外部メモリには DRAM を想定し，古典的な Activate-Read/Write-Precharge のコマンドシーケンスによりアクセスされるとみなす．よって，メモリへアクセス要求が出されてから最初のデータが読み出し，または書き込まれるまでのレイテンシは 3 サイクルに設定している．変更するシステムのパラメータは次のとおりである．下線太字の構成は従来の設計を示す．

- キャッシュ
  - way 数：ダイレクトマップ，**2way**，4way
  - ブロックサイズ：**64Byte**，128Byte
- 内部バス幅：**4Byte**，8Byte，16Byte

キャッシュヒット時間の増大を避けるために，キャッシュのインデックスは一般に仮想アドレスのページ内オフセットで定まるようにする．MIPS R3000 では 20 ビットのページ番号を用いるため，キャッシュのインデックスには最大で 12 ビットが利用できる．**3.3** で説明したように，Geyser は 4KB の 2way セットアソシアティブの命令キャッシュとデータキャッシュをもち，ブロックサイズは 64Byte である．つまり，way あたりのブロック数は 64 であり，6 ビットをブロックのインデックス，残りの 6 ビットをブロック内オフセットとして扱う．128 Byte のブロックサイズを検討する場合も同様に 12 ビットを超えないようにする．したがって，64Byte の場合と比較してブロック数は半分となり，way 数が同じであれば全体のキャッシュ容量は変わらない．また，ルータによるブロック転送時のフリット数もキャッシュブロックのサイズと同一であるとする．**3.2** で示したように，パケットは可変フリット数で構成可能であるため，フリットフォーマットの変更は不要であるが，最大パケット長が 17 フリットから 33 フリットへと約 2 倍になるため，各仮想チャネルバッファの深さを合わせて大きくする必要がある．

更に，内部のバス幅を従来の 4Byte (1 ワード) から変更する場合，TCI による転送能力も考慮する必要がある．**3.2** で説明したとおり，現在の TCI-IP では 1 クロックサイクルに 35bit のデータを転送する．したがって，従来の TCI リンクでは 4 倍のデータ幅に対応するのは困難である．そこで，バス幅の拡張に合わせて図 7 に示した TCI リンクを検討する．1 レーンあたり 1 フリットのデータ転送を行い，図 7 (a) は従来の IP に実装されているものである．バス幅が 2 ワードの場合は 2 レーン構成を利用し，バス幅が 4 ワードの場合は 4 レーン構成を利用する．

5.3.1 性 能 評 価

図 8 に各システム構成で対象のアプリケーションを実行した場合の実行時間の比較を示す．アクセラレータによるアプリケーション高速化の度合いを確認するために，従来構成 (2way，4Byte バス，64Byte ブロック) の Geyser 単体で実行した場合の実行時間で正規化している．ただし，AlextNet では SNACC を利用した場合と Geyser 単体で実行時間の差が大きいため，二つのグラフ (図 8 (b) と図 8 (c)) に分けて示す．

JPEG のエンコードでは従来構成の Geyser 単体で実行に 213,056 サイクルを要し，50MHz の動作周波数の場合実行時間は 4.26msec となる．Geyser 単体の場合，ダイレクトマップキャッシュではキャッシュミスが大きく増加し，結果的に 2way の場合と比較して実行時間が 10% 程度増加している．しかし，2way から 4way へ増やした場合の効果は非常に小さいとわかる．

CC-SOTB2 をはじめとする CGRA はプログラム中の計算負荷が高いループ部分をアクセラレーションの対象とする．一方で，分岐処理などを苦手とするため，CC-SOTB2 で処理のできない部分は Geyser で実行する必要がある．したがって，CC-SOTB2 に処理用データを送る前の処理と結果を回収した後の処理も多く存在する．これが理由で，Geyser 単体の場合と同様に CC-SOTB2 を利用した場合でも Geyser 側のキャッシュ構成が全体の実行時間に影響を与えている．今回評価に利用した MiBench のコードは逐次処理向けに記述されたものであり，2 枚の CC-SOTB2 で効率的に処理できる部分は離散コサイン変換とそれに続く量子化のパートに限られることがわかった．よって，積層チップを増やしたことによるデータ転送が増加した影響もあり劇的な性能向上は得られておらず，従来構成の Geyser をホストにした場合，CC-SOTB2 を一枚利用して 1.34 倍，2 枚利用して 1.54 倍の性能向上にとどまる．バス幅を 4 倍の 16Byte にしても性能向上はそれぞれ 1.64 倍，1.84 倍である．しかし，CGRA のもつ演算処理性能を最大限に引き出すための専用プログラミングモデルによって更なる性能向上が期待できる．これを解決するために現在，CGRA 向けのソフトウェア開発環境の検討 [29] が行われている．

AlexNet の FC7 層は入力数，出力数ともに 4096 の全結合層であり，従来構成の Geyser 単体では 230,063,425 サイクルを要し，50MHz の動作周波数の場合実行時間は 4.60sec となる．Geyser 単体ではダイレクトマップにした場合の性能悪化が著しい．64Byte のキャッシュブロックでは約 5 倍の実行時間となり，128Byte のキャッシュブロックでは更にキャッシュミス率が増えるため約 7 倍の実行時間となっている．

一方で，SNACC を利用した場合の実行時間はキャッシュの way 数に影響を受けていない．これは，CC-

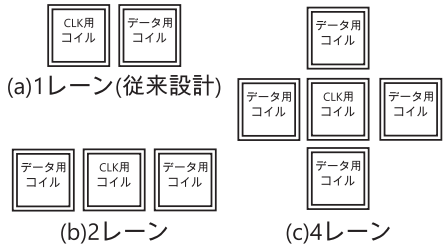
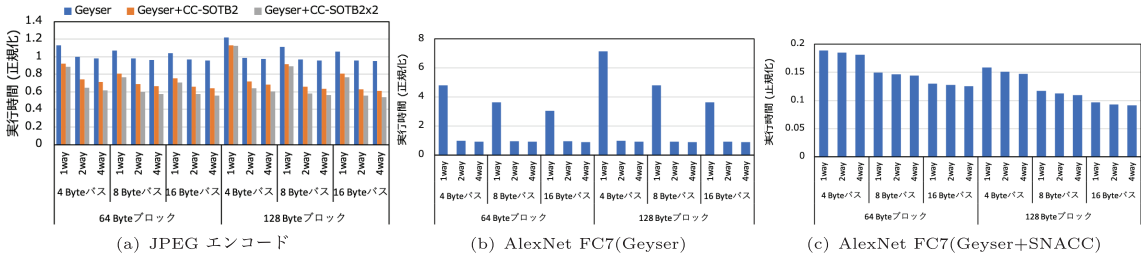
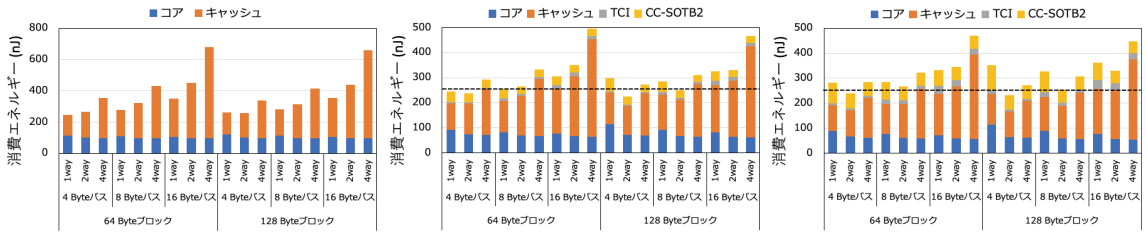


図 7　複数レーン TCI リンクの検討



(a) JPEG エンコード　(b) AlexNet FC7(Geyser)　(c) AlexNet FC7(Geyser+SNACC)

図 8　各システム構成における実行時間の比較



(a) Geyser 単体　(b) Geyser+CC-SOTB2　(c) Geyser+CC-SOTB2x2

図 9　各システム構成における消費エネルギー (JPEG エンコード)

SOTB2 の場合と異なりほとんどの処理を SNACC 側で実行可能で，Geyser はデータの転送や SNACC のコアでの計算開始などの制御を行うだけだからである．SNACC のメモリはダブルバッファ構成となっており，コアで計算を行っている間に次のデータを転送することが可能である．データ転送のほぼ全てが全結合層の重みデータであり，これに全体の実行時間の 90% 以上を要する．したがって，SNACC を 2 枚に増やしても，データ転送の時間が 2 倍になるだけで速度向上を望むことができない．しかし，SNACC 1 枚でも十分な性能向上が得られており，従来構成の Geyser でも約 5.5 倍の性能向上が得られている．前述のとおりデータ転送が支配的であり，キャッシュミスの影響が小さいためキャッシュブロックとバス幅の増加が更なる性能向上に貢献している．キャッシュブロックを大きくするとバスやメモリのアクセスレイテンシの影響が小さくなるためである．128Byte のキャッシュブロックで，16Byte のバス幅にすることで性能向上は約 11 倍となる．

5.3.2　エネルギー評価

次に，各システム構成の消費エネルギーを評価する．エネルギーは以下のモデルを用いて推定する．

$$E_{total} = E_{cpu} + E_{accelerator} \tag{1}$$

$$E_{cpu} = E_{core} + E_{cache} + E_{TCI} \tag{2}$$

$$E_{core} = P_{core} \times T_{exec} \tag{3}$$

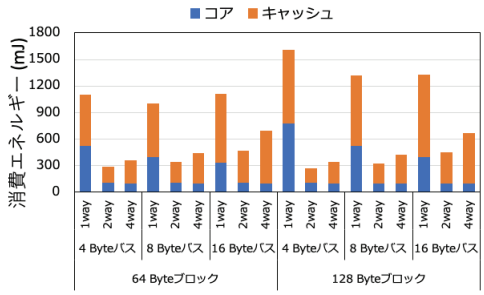
$$E_{cache} = \alpha P_{cache,idle} \times T_{exec} + E_{cache,access} \times C_{access} \tag{4}$$

$$E_{TCI} = P_{TCI,idle} \times N_{lane} \times T_{exec} + E_{flit} \times C_{flit} \tag{5}$$

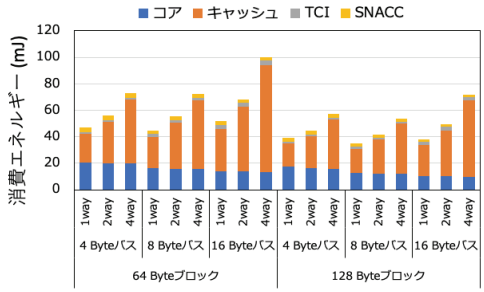
システム全体のエネルギー E_{total} はプロセッサのエネルギー E_{cpu} とアクセラレータのエネルギー $E_{accelerator}$ の和である．E_{cpu} は大きくコア (E_{core})，キャッシュ (E_{cache})，TCI (E_{TCI}) のエネルギーに分けられる．ルータのエネルギーは十分小さかったため，コアのエネルギーにまとめている．P_{core} はコアで消費される電力で，本評価では常に一定とする．$P_{cache,idle}$，$P_{TCI,idle}$ はキャッシュ及び TCI で定常的に消費される電力であり，クロックツリーでのダイナミック電力やスタティック電力を含む．ただし，$P_{cache,idle}$ は従来の Geyser の設計におけるものであるため，キャッシュ構成の違いによるオーバーヘッド α によってスケールされる．このオーバーヘッドは CACTI 7 [30] を用いて各キャッシュ構成の電力をシミュレーションし，従来構成と比較して何倍の電力消費となるかを評価することで得られる．$P_{TCI,idle}$ は TCI リンク一つあたりの電力であるため，リンク数 N_{lane} の分だけ加算する．T_{exec} は CubeSim によってシミュレーションされた全体の実行時間である．更に，キャッシュの 1 ワードにアクセスするのに消費するエネルギー $E_{cache,access}$ とアクセス数 C_{access} の積，TCI で 1 フリット送受信するのにかかるエネルギー E_{flit} と転送フリット数 C_{flit} の積を加える．本評価ではアクセラレータ自体の設計は固定であるため，TCI のレーン数変更の影響を除き $E_{accelerator}$ は電力と動作時間の積でも算出される．

各モジュールの電力やエネルギーは試作チップの電力測定の結果 [8], [31], [32] を用いる．ただし，Geyser の各モジュールの電力内訳は Synopsys 社 IC Compiler の電力レポートを元に決定する．$P_{TCI,idle}$ 及び E_{flit} は [33] で報告されている SPICE シミュレーションによる値を利用する．JPEG エンコードの消費エネルギーを図 9 に AlexNet FC7 層の消費エネルギーを図 10 に示す．

CC-SOTB2 を利用する構成の評価結果 (図 9 (b) と図 9 (c)) における破線は従来構成の Geyser 単体での消費エネルギーである．CC-SOTB2 を利用した場合



(a) Geyser 単体



(b) Geyser+SNACC

図 10　各システム構成における消費エネルギー (AlexNet FC7)

の高速化は従来構成では 1.3 倍程度あったため，追加チップの CC-SOTB2 や TCI の電力も加わり削減されるエネルギーは 10% 程度となる．また，バス幅やキャッシュ way 数を増やしたほとんどの構成は従来構成の Geyser よりもエネルギーが悪化している．結果的に，CC-SOTB2 を用いる場合は，利用枚数にかかわらず 2way 構成，4Byte バスのままキャッシュサイズを 128Byte にするのが最善であるものの，構成変更によるエネルギーの削減効果は 5% 程度である．

AlexNet を Geyser 単体で実行する場合は性能評価で確認したようにダイレクトマップでは著しく実行時間が増加し，消費エネルギーが大幅に増加している．対して，SNACC を用いる場合はキャッシュの影響が小さいため，最も電力が小さいダイレクトマップキャッシュがどの構成においても最も少ない消費エネルギーとなっている．前述のとおりデータ転送が支配的であったことから，SNACC を用いる場合は 8Byte のバス幅で 128Byte のキャッシュブロックのダイレクトマップキャッシュにした場合が最も消費エネルギーが小さく，従来構成の Geyser 単体と比較して約 87% のエネルギー削減が得られる．また，従来構成の Geyser+SNACC と比較しても約 38% のエネルギー削減が得られる．しかし，2way のキャッシュにしても約 84% のエネルギー削減を得ることができ，OS をはじめとする SNACC 制御以外の処理も動作させることを考慮するとこちらの構成の方が妥当であると言える．

6. むすび

本研究では誘導結合 TCI を用いた 3 次元積層システム向けのサイクルアキュレートシミュレータ CubeSim を開発した．CubeSim を用いてホストプロセッサの Geyser と CC-SOTB2 を組み合わせたシステムで JPEG エンコードを，SNACC を組み合わせたシステムで AlexNet の FC7 層を処理した場合の実行時間の見積もりを行った．評価の結果，CubeSim は RTL シミュレーションと比べ最大で約 234 倍の高速化を達成した．CubeSim ではバス幅やキャッシュの構成がパラメータ化されており，システムの構成変更が実行時間に与える影響を定量的に評価することができるようになった．更に，この評価結果を用いて様々なシステム構成のエネルギー評価を行い，開発した CubeSim をシステムの設計探索に利用できることを示した．

謝辞　本研究は，JSPS 科研費 (B) ビルディングブロック型計算システムにおけるチップブリッジを用いた積層方式 (18H03125)，JSPS 科研費 3 次元積層技術を応用した粗粒度再構成可能デバイスの研究 (19J21493)，及び，科学技術振興機構戦略的研究推進事業 (JST)，CREST，JPMJCR19K1 の支援を受けたものである．また，本研究は東京大学大規模集積システム設計教育研究センターを通しシノプシス株式会社並びにケイデンス株式会社の協力で行われたものである．関係者の皆様に感謝致します．

文　献

[1] J.T. Pawlowski, “Hybrid memory cube (HMC),” 2011 IEEE Hot chips 23 symposium (HCS) IEEE, pp.1–24, 2011.

[2] J. Standard, “High bandwidth memory (hbm) dram,” JESD235, 2013.

[3] X. Hu, D. Stow, and Y. Xie, “Die stacking is happening,” IEEE Micro, vol.38, no.1, pp.22–28, 2018.

[4] M.-F. Chen, F.-C. Chen, W.-C. Chiou, and C. Doug, “System on Integrated Chips (SoIC (TM) for 3D Heterogeneous Integration,” 2019 IEEE 69th Electronic Components and Technology Conference (ECTC) IEEE, pp.594–599, 2019.

[5] D. Ingerly, S. Amin, L. Aryasomayajula, A. Balankutty, D. Borst, A. Chandra, K. Cheemalapati, C. Cook, R. Criss, K. Enamul, et al., “Foveros: 3D Integration and the use of Face-to-Face Chip Stacking for Logic Devices,” 2019 IEEE International Electron Devices Meeting (IEDM) IEEE, pp.19.6.1–19.6.4, 2019.

[6] Y. Take, H. Matsutani, D. Sasaki, M. Koibuchi, T. Kuroda, and H. Amano, “3D NoC with inductive-coupling links for building-block

SiPs," IEEE Trans. Computers, vol.63, no.3, pp.748–763, 2012.
[7] Y. Koizumi, N. Miura, E. Sasaki, Y. Take, H. Matsutani, T. Kuroda, H. Amano, R. Sakamoto, M. Namiki, K. Usami, et al., "A scalable 3D heterogeneous multi-core processor with inductive-coupling thruchip interface," IEEE Micro, pp.6–15, 2013.
[8] S. Terashima, T. Kojima, H. Okuhara, K. Musha, H. Amano, R. Sakamoto, M. Kondo, and M. Namiki, "A Preliminary evaluation of building block computing systems," 2019 IEEE 13th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC) IEEE, pp.312–319, 2019.
[9] A. Akram and L. Sawalha, "A survey of computer architecture simulation techniques and tools," IEEE Access, vol.7, pp.78120–78145, 2019.
[10] N. Binkert, B. Beckmann, G. Black, S.K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D.R. Hower, T. Krishna, S. Sardashti, et al., "The gem5 simulator," ACM SIGARCH computer architecture news, vol.39, no.2, pp.1–7, 2011.
[11] F. Bellard, "QEMU, a fast and portable dynamic translator.," USENIX Annual Technical Conference, FREENIX Track, vol.41, p.46, 2005.
[12] J. Larus, "Spim: A mips32 simulator," Computer Science Department, University of Wisconsin-Madison, 2011. http://pages.cs.wisc.edu/~larus/spim.html, 参照 Feb. 5, 2021.
[13] D. Aarno and J. Engblom, Software and system development using virtual platforms: full-system simulation with wind river simics, Morgan Kaufmann, 2014.
[14] B. Gaeke, "The VMIPS Project, Version 1.5.1," http://vmips.sourceforge.net/vmips/, 2019.
[15] R. Ubal, J. Sahuquillo, S. Petit, and P. Lopez, "Multi2sim: A simulation framework to evaluate multicore-multithreaded processors," 19th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD'07) IEEE, pp.62–68, 2007.
[16] N. Agarwal, T. Krishna, L.-S. Peh, and N.K. Jha, "GARNET: A detailed on-chip network model inside a full-system simulator," Performance Analysis of Systems and Software, 2009. ISPASS 2009. IEEE International Symposium on IEEE, pp.33–42, 2009.
[17] L. Zhao, D. Ikebuchi, Y. Saito, M. Kamata, N. Seki, Y. Kojima, H. Amano, S. Koyama, T. Hashida, Y. Umahashi, et al., "Geyser-2: The second prototype CPU with fine-grained run-time power gating," Proc. 16th Asia and South Pacific Design Automation Conference, IEEE Press, pp.87–88, 2011.
[18] N. Ozaki, Y. Yasuda, M. Izawa, Y. Saito, D. Ikebuchi, H. Amano, H. Nakamura, K. Usami, M. Namiki, and M. Kondo, "Cool Mega-Arrays: Ultralow-Power Reconfigurable Accelerator Chips," IEEE Micro, vol.31, no.6, pp.6–18, Nov. 2011.
[19] 松下悠亮，増山滉一朗，野村明生，門本淳一郎，四手井綱章，黒田忠広，天野英晴，"誘導結合ワイヤレスチップ間接続の IP 化," 信学技報，CPSY2016-58, 2016.
[20] R. Sakamoto, R. Takata, J. Ishii, M. Kondo, H. Nakamura, T. Ohkubo, T. Kojima, and H. Amano, "The design and implementation of scalable deep neural network accelerator cores," 2017 IEEE 11th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC) IEEE, pp.13–20, 2017.
[21] Y. Tokusashi, H. Matsutani, and H. Amano, "Key-value Store Chip Design for Low Power Consumption," 2019 IEEE Symposium in Low-Power and High-Speed Chips (COOL CHIPS) IEEE, pp.1–3, 2019.
[22] A. Nomura, H. Matsutani, T. Kuroda, J. Kadomoto, Y. Matsushita, and H. Amano, "Vertical packet switching elevator network using inductive coupling ThruChip interface," 2016 Fourth International Symposium on Computing and Networking (CANDAR) IEEE, pp.195–201, 2016.
[23] N. Miura, H. Ishikuro, T. Sakurai, and T. Kuroda, "A 0.14 pJ/b inductive-coupling inter-chip data transceiver with digitally-controlled precise pulse shaping," Solid-State Circuits Conference, 2007. ISSCC 2007. Digest of Technical Papers. IEEE International IEEE, pp.358–608, 2007.
[24] 並木美太郎，小柴篤史，濱田槙亮，大城研治，天野英晴，"TCI 結合による計算機システム向けビルディングブロック OS について，" 第 80 回全国大会講演論文集，vol.2018, no.1, pp.19–20, 2018.
[25] Y. Hara, H. Tomiyama, S. Honda, and H. Takada, "Proposal and quantitative analysis of the CHStone benchmark program suite for practical C-based high-level synthesis," J. Information Processing, vol.17, pp.242–254, 2009.
[26] T. Kojima, N. Ando, Y. Matshushita, H. Okuhara, N.A.V. Doan, and H. Amano, "Real chip evaluation of a low power CGRA with optimized application mapping," Proc. 9th International Symposium on Highly-Efficient Accelerators and Reconfigurable Technologies ACM, p.13, 2018.
[27] M.R. Guthaus, J.S. Ringenberg, D. Ernst, T.M. Austin, T. Mudge, and R.B. Brown, "MiBench: A free, commercially representative embedded benchmark suite," Workload Characterization, 2001. WWC-4. 2001 IEEE International Workshop on IEEE, pp.3–14, 2001.
[28] A. Krizhevsky, I. Sutskever, and G.E. Hinton, "Imagenet classification with deep convolutional neural networks," Proc. 25th International Conference on Neural Information Processing Systems - Volume 1, pp.1097–1105, NIPS'12, Curran Associates Inc., USA, 2012.
[29] 大和田彩夏，小島拓也，天野英晴，"LLVM を用いた CGRA 向けアプリケーション開発環境の構築と評価，" 信学技報，CPSY2019-109, 2020.
[30] R. Balasubramonian, A.B. Kahng, N. Muralimanohar, A. Shafiee, and V. Srinivas, "CACTI 7: New tools for interconnect exploration in innovative off-chip memories," ACM Transactions on Architecture and Code Optimization (TACO), vol.14, no.2, pp.1–25, 2017.
[31] R. Tomura, T. Kojima, H. Amano, R. Sakamoto, and M. Kondo, "A Real Chip Evaluation of a CNN Accelerator SNACC," Proc. of SASIMI 2019, pp.62–67, 2019.
[32] T. Kojima, N. Ando, H. Okuhara, and H. Amano, "Glitch-aware variable pipeline optimization for CGRAs," 2017 International Conference on ReConFigurable Computing and FPGAs (ReConFig), pp.1–6, Dec. 2017.
[33] H. Zhang, H. Matsutani, Y. Take, T. Kuroda, and H. Amano, "Vertical link on/off regulations for inductive-coupling based wireless 3-D NoCs," IEICE Trans. Information and Systems, vol.96, no.12, pp.2753–2764, 2013.

（2020 年 6 月 5 日受付，8 月 13 日再受付，
2021 年 1 月 6 日早期公開）

小島　拓也　（学生員）

平 29 慶大理工学部卒．平 31 同大大学院理工学研究科開放環境科学専攻修士課程了．現在，同大大学院理工学研究科開放環境科学専攻博士課程在学中．粗粒度再構成可能デバイス及び 3 次元積層型 LSI の研究に従事．

池添　赳治

平 30 慶大理工学部卒．令 2 同大大学院理工学研究科開放環境科学専攻修士課程了．同大学在学中不揮発性メモリ技術の研究に従事．

天野　英晴　（正員：フェロー）

昭 56 慶大学工・電気卒．昭 61 同大大学院理工学研究科電気工学専攻博士課程了．現在，慶應義塾大学理工学部情報工学科教授．工博．計算機アーキテクチャの研究に従事．