高性能・高密度実装・低消費電力を実現するスーパーコンピュータ「富岳」のCPU A64FX

岡崎　亮平　　田端　猛一　　坂下　聡太　　北村　健一　　高木　紀子　　坂田　英樹
石橋　武史　　中村　武夫　　安島　雄一郎

あらまし

A64FXはスーパーコンピュータ「富岳」のプロセッサーとして開発された。半導体にはTSMCの7 nm CMOSプロセスを採用し，高密度化のためにTofuインターコネクトDコントローラーとPCI Expressコントローラーを統合させ，パッケージ内に高帯域な3次元積層メモリーを搭載している。

A64FXは，実績のある富士通の高性能なマイクロアーキテクチャーを受け継ぎながら，ソフトウェア開発環境を向上させるために，Armアーキテクチャーを採用する。更に，Arm社のリードパートナーとしてSVE（Scalable Vector Extension）の仕様策定に取り組み，その成果を採用した。

本稿ではA64FXの概要と高性能なマイクロアーキテクチャー，高密度実装を実現するアーキテクチャー，低消費電力設計について述べる。

1. まえがき

A64FX（図-1）は，スーパーコンピュータ「富岳」（以下，「富岳」）のプロセッサーとして開発された。「富岳」は158,976個のプロセッサーを搭載する[1]。そのプロセッサーには，高性能，高密度実装，低消費電力がいずれも高いレベルで求められる。更に，ソフトウェア開発環境も重要である。

本稿では，A64FXの概要と高性能なマイクロアーキテクチャー，高密度実装を実現するアーキテクチャーと低消費電力設計について述べる。

2. A64FXの概要とArmアーキテクチャーの採用

A64FXでは，スーパーコンピュータ「京」（以下，「京」）を含め，富士通がこれまで開発してきた各種プロセッサーのマイクロアーキテクチャーをベースにして，アプリケーションをより高速に実行できることを目標に開発を進めることとした。

アプリケーションの高速化を実現するために，各種アプリケーションの解析を行うとともに，各種ブロックの構成の見直し，資源の最適化，新規回路の追加，メモリー部品の選択と最適化，OS動作を含めたプロセッサー全体の最適化を行った。また，汎用CPUであるA64FXを搭載したシステムが，GPUなどを搭載するシステムと同等の電力あたり性能を実現できることを目標に，アーキテクチャーレベルからデバイスレベルまで，広い階層にわたって省電力のための工夫を行った。

A64FX CPUのブロック図を図-2に示す[2]。13個のコア（12個を計算コアとして使用し，1個をアシスタントコアとして使用），2次キャッシュ，メモリーコントローラーで構成するCore Memory Group（CMG）という四つのグループと，TofuインターコネクトD（以降，TofuD）[3]インターフェイス，PCI Express（以下，PCIe）インターフェイスをリングバスのNetwork on Chip（NoC）で接続している。

A64FXを開発するに当たり，スーパーコンピュータ「富岳」では2012年から本格共用稼働した「京」

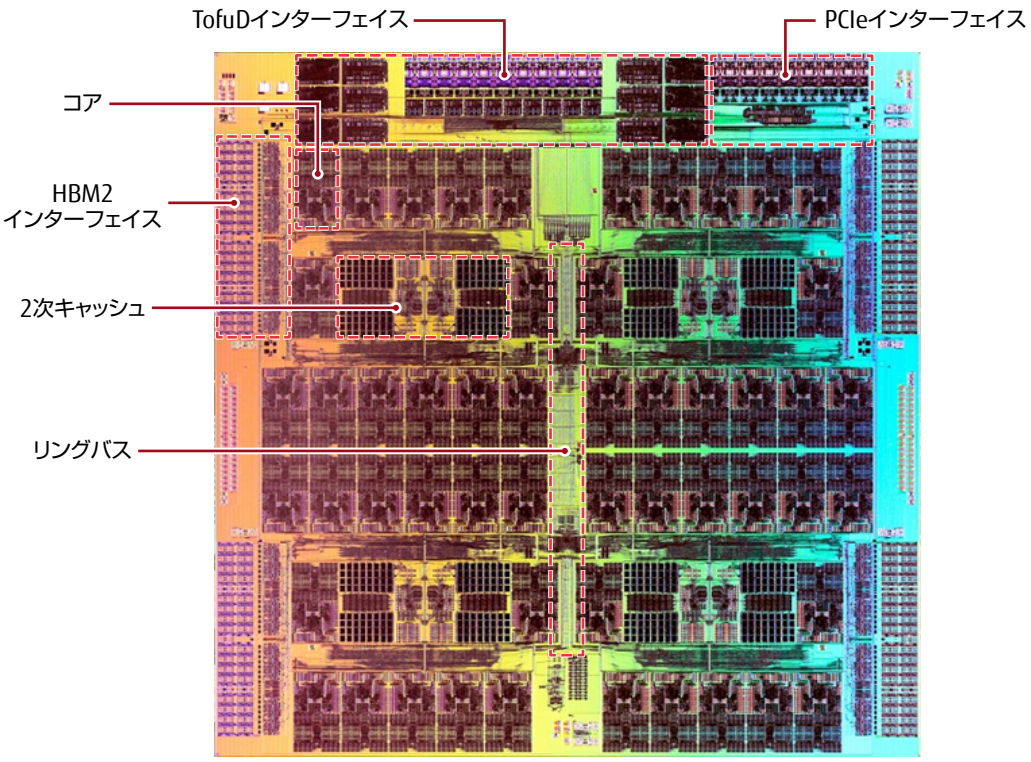

図-1　A64FX CPUのダイ写真

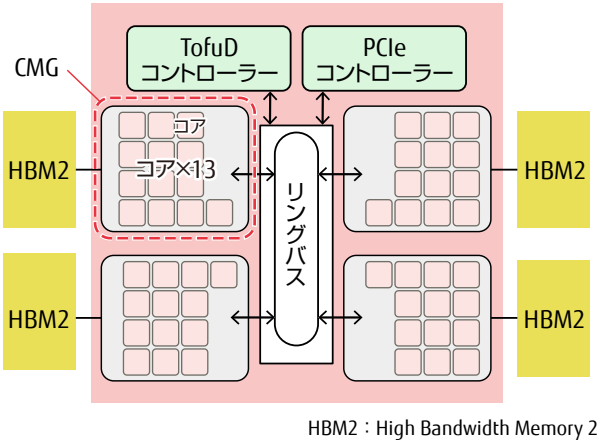

図-2　A64FX CPUのブロック図

よりも幅広いソフトウェア開発者に受け入れられることと，最新のソフトウェアが使える環境を整えることを目標としてArmアーキテクチャーを採用した。Armアーキテクチャーは，Arm Limited（以下，Arm社）が開発した命令セットであり，スマートフォンや組み込み機器などのソフトウェア開発に幅広く用いられている。近年では，サーバ分野で標準の64 bitアーキテクチャーへの拡張や，サーバ向けハイパーバイザ拡張機能などの新規追加によって，サーバ向けArmプロセッサーも登場している。これらの背景から，Armはサーバ分野への広がりが期待されている。

ArmアーキテクチャーをHPC（High-Performance Computing）向けプロセッサーとして提供していく上で課題となったのは，Armアーキテクチャーが独自で持っていたSIMD（Single Instruction Multiple Data）拡張の部分である。このSIMD拡張はAdvanced SIMDと呼ばれ，組み込みなどの用途でメディアプロセッシングやDSP（Digital Signal Processing）などを加速するために用いられている。SIMD長は「京」と同じ128 bitである。これは，現在のHPC向けCPUのトレンドである256 bit，512 bitと比べて短く，1コア当たりの演算性能を向上させるには不向きな構成であった。また，富士通のこれまでのHPC開発の経験から，HPCアプリケーションで有用な命令も不足していた。

そのため，富士通はArm社と協業することにより，科学技術計算，AIを含むHPCアプリケーションを高速に実行できるSVE（Scalable Vector Extension）の策定にリードパートナーとして貢献し，その成果をA64FXに採用した。

3.　高性能なマイクロアーキテクチャー

A64FXでは，ユーザーアプリをより高速に実行できることを目標に，「富岳」専用のマイクロアーキテクチャーの開発を行った。本章では，マイクロアーキテクチャーの概要，およびそれを実現する要素技術について述べる。

3.1　マイクロアーキテクチャー概要

A64FXのパイプラインを図-3に示す[2]。コアは，命令制御部，演算処理部，1次キャッシュ部からなる。命令制御部は，命令フェッチ，命令デコード，命令のアウトオブオーダ処理制御，そして命令完了の制御を行う。

演算部は，固定小数点演算を行う二つの演算器（EXA/EXB），アドレス計算と単純な固定小数点演算を行う二つの演算器（アドレス計算時はEAGA/EAGBと呼び，固定小数点演算を行う場合にはEXC/EXDと呼ぶ），SVE命令の演算を実行する二つの浮動小数点演算器（FLA/FLB），およびプレディケート演算を実行する一つのプレディケート演算器（PRX）を備える。浮動小数点演算器はどちらも512 bit SIMD構成をとり，1サイクルごとに浮動小数点の積和演算を実行できる。したがって，各計算コアでは1サイクル当たり32個，チップ内の全計算コアでは1サイクル当たり1,536個の倍精度浮動小数点演算が実行可能である。単精度浮動小数点演算あるいは半精度浮動小数点演算であれば，それぞれ倍精度浮動小数点演算の2倍，4倍の演算が可能である。動作周波数は搭載されるシステムによって異なるが，1.8 GHz/2 GHz/2.2 GHzで動作する。

1次キャッシュ部は，ロード/ストア命令を処理する。コアごとに64 KiBの命令キャッシュと64 KiBのデータキャッシュをそれぞれ有する。データキャッシュは，二つ同時にロードアクセスが可能な構成であり，64バイトのSIMDロードを二つ，または64バイトのSIMDストアを一つ実行する。

2次キャッシュ部は，一つのCMG当たり8 MiBの

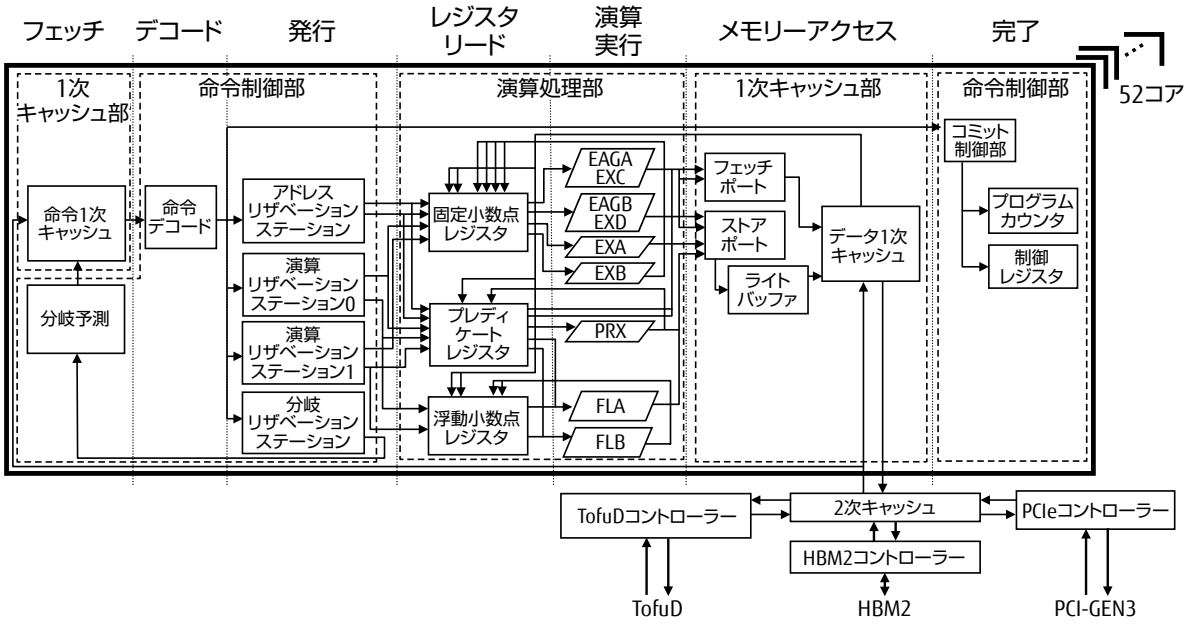

図-3　A64FXのパイプライン図

ユニファイドキャッシュを有しており，アシスタントコアを含む13のコアで共有される。

3.2 実績のあるマイクロアーキテクチャーとリソースの最適化

A64FXでは，富士通がこれまで開発してきたメインフレームやUNIXサーバ，およびスーパーコンピュータ「京」で採用した高性能・高信頼なマイクロアーキテクチャーをベースに，各種ハードウェアのリソースの最適化を実施した。

特に，性能の指標として重要なROB（リオーダバッファ）やリザベーションステーションなどのキューの数については，キューの開放タイミングを命令実行時に判断して開放を早める制御を採用している。これによって，キューのエントリー数を無駄に多くすることなく命令実行性能を確保することができ，論理回路の増加を抑えることでチップ面積を小さくすることに貢献している。

3.3 分岐予測回路

様々なアプリケーションで最適な分岐予測ができるように，幾つかの分岐予測回路を採用した。

例えば，複雑な命令構造のプログラムでも高精度に分岐予測を行えるように，Piecewise Linear方式のアルゴリズムを用いて分岐予測を行う回路を採用した。これにより，長い命令実行の履歴を基に分岐予測を行うことができ，高い予測精度を実現できる。

また，単純なループなどのプログラム構造を検出して分岐予測を行う回路も採用している。プログラムがループしている間にループ命令列のバッファリングを行うことで，命令フェッチ部と他の分岐予測回路の動作を止めて，電力を抑えている。

3.4 Virtual Fetch/Store Port回路

ロード/ストア系命令で使用するFetch/Store Portは，パイプラインの後段に位置するキューである。そのため，リソース管理をしているデコーダー部に開放通知を行ってから開放したエントリーが実際に使われるまで数サイクルが必要である。リソースフルになったときには，Fetch/Store Portを使わない命令も含めて命令デコードが止まってしまうため，Fetch/Store Port数の使用が多いSVE命令に対して，何らかの対応が必要だった。

この課題に対して，A64FXではVirtual Fetch/Store Port機能をサポートすることで，最小限の回路規模でFetch/Store Portの使用効率を高めて，性能を向上させることとした。

従来はFetch/Store Portのリソース管理を命令

デコーダーで行っていた。しかしA64FXでは，命令デコーダーでは実際のFetch/Store Port数より多い仮想的なFetch/Store Port（Virtual Fetch/Store Port）を割り当て，ロード/ストア系命令で使用するリザベーションステーションでFetch/Store Portのリソース管理を行うようにした。これによって，Fetch/Store Portが全て使用されているときにも命令デコードが止まらなくなるため，Fetch/Store Portのエントリー数を増やしたのと同等の効果を得ることができた。

3.5 多様なアクセスパターンに対応したL1キャッシュ

512 bit SIMDの効率を最大化するためには，レジスタへデータを転送するロード命令で，L1キャッシュのアクセススループットを維持することが重要である。アドレスが512 bit境界になっていないロード命令をアドレスの順に連続して実行した場合，数命令に一度キャッシュラインをまたぐアクセスが発生する。このケースで性能低下させないために，A64FXのL1キャッシュでは2本のリードポートのそれぞれが連続する二つのキャッシュラインに常時アクセスが可能な構成としている。これにより，ロード命令がキャッシュラインをまたいでアクセスする場合であっても，毎サイクル512 bit×2のスループットを維持している。

3.6 Gather Load命令の高速化

Gather Load命令は，非連続な複数要素のデータをメモリーから読み出し，一つのレジスタに書き込む命令である。非連続なデータではあるが，HPCアプリケーションでは，複数要素が近いアドレスにアクセスするなど，データに局所性があることが分かっている。この特徴を踏まえ，A64FXではCombined Gatherという高速化機構を導入した。Combined Gather機構では，Gather Load命令を1要素ずつ処理するのではなく，2要素からなるグループに分解する。そして，同一グループに属する要素が同一の128 Byte境界内のメモリーにアクセスする場合は，一度のキャッシュアクセスで処理を完了する。これにより，1要素ずつ処理した場合と比較して，2倍のスループット性能を実現した。

3.7 プリフェッチ機構

主記憶装置や下位階層キャッシュへのアクセスには多くの時間を要するため，プログラムの性能向上のためにはアクセス時間を隠蔽するためのプリフェッチが重要である。プリフェッチの実現方法として，大きくソフトウェアによる手法と，ハードウェアによる手法とがある。

(1) ソフトウェアプリフェッチ

A64FXは，通常のプリフェッチ命令であるARMv8プリフェッチ命令に加え，SVEのSVE Contiguousプリフェッチ命令，SVE Gatherプリフェッチ命令をサポートした。これにより，複数キャッシュラインにまたがるプリフェッチも1命令で発行する。

(2) ハードウェアプリフェッチ

A64FXのハードウェアプリフェッチ機構は，「Stream detectモード」と「Prefetch injectionモード」の二つのモードを備える。

Stream detectモードは，連続アクセスに対するプリフェッチを発行するモードである。Prefetch Queueと呼ばれる専用の機構を用いてメモリーアクセスを監視し，連続アクセスストリームを検出するとアドレスの連続する方向にプリフェッチを発行する。

Prefetch injectionモードは，メモリーアクセスに対して一定距離離れたアドレスへのプリフェッチを発行するモードである。このモードを利用するプログラムは，事前にプリフェッチアドレスに関する情報を専用のプリフェッチ制御用レジスタに設定しておくことで，一定距離離れたアドレスをアクセスするストライドアクセスに対応したプリフェッチを発行可能としている。

4. 高密度アーキテクチャー

A64FXは高密度化のために，各種コントローラーを統合したSoC（System-on-a-chip）であるだけでなく，パッケージ内に四つの高帯域メモリー（HBM2）を搭載する。四つのCMGはそれぞれHBM2と接続され，低レイテンシと高帯域を確保する。

4.1 CMG構成とccNUMA

図-4にCMG構成と接続図を示す［4］。CMGは，13個のコアとそれらのコアが共有する2次キャッシュ

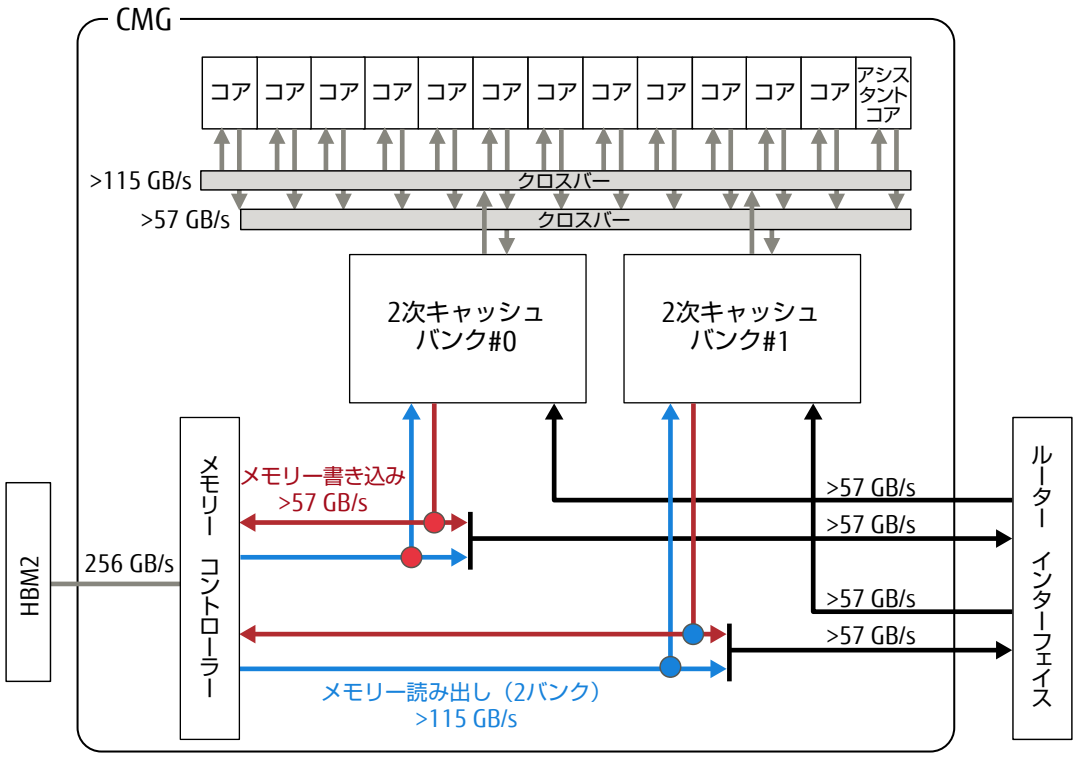

図-4　CMG構成と接続図

部，およびメモリーコントローラーで構成される。

2次キャッシュの容量はCMG当たり8 MiBであり，二つの2次キャッシュバンクと13個のコアはクロスバーで接続されている。また，ソフトウェアがCMGをNUMA（Non-Uniform Memory Access）ノードとして扱うことができるように，2次キャッシュ部はチップ全体のキャッシュコヒーレントを保証するための機能を持っている。

A64FXにおけるキャッシュコヒーレントは，一般的なhome agent機構を用いず，2次キャッシュパイプラインによって一元管理されている。2次キャッシュパイプラインは，前半をローカル・パイプライン，後半をグローバル・パイプラインと呼ぶ1本のパイプラインで構成している。CMG間のキャッシュコヒーレントを管理するためのディレクトリ情報はTAGD（TAG Directory）と呼ばれる部分に格納され，グローバル・パイプラインでアクセスされる。CMG内に閉じたコヒーレント管理が可能な場合はローカル・パイプラインの終端でコアへの応答を，そうでない場合はそれに続くグローバル・パイプラインの終端でCMG間のコヒーレント制御を，それぞれ開始する。この構成により，ハードウェアリソースの削減を行うとともに，低レイテンシなccNUMA（cache-coherent NUMA）システムを実現している。

4.2 SoCアーキテクチャー

図-5にCMG間の接続図を示す [3]。四つのCMG，およびTofuD/PCIeコントローラー，割り込みコントローラーは，2本のリングバスと6本のリングストップで接続する。A64FXでは，更に隣接するCMG間を接続するためのCMGインターコネクトパスを設けた，独自ネットワークオンチップトポロジを採用した。リングバスおよびCMGインターコネクトパスのスループットは，それぞれ1本当たり115 GB/s以上である。

CMGインターコネクトパスを設けることにより，TofuDネットワークやI/Oとのデータ転送，および割り込みコントローラーからの割り込み要求などによるリングバス使用の影響を避けるとともに，クロスバーよりブロック間の接続数の少ない構成となり，隣接するCMG間のスループット性能を担保することができる。

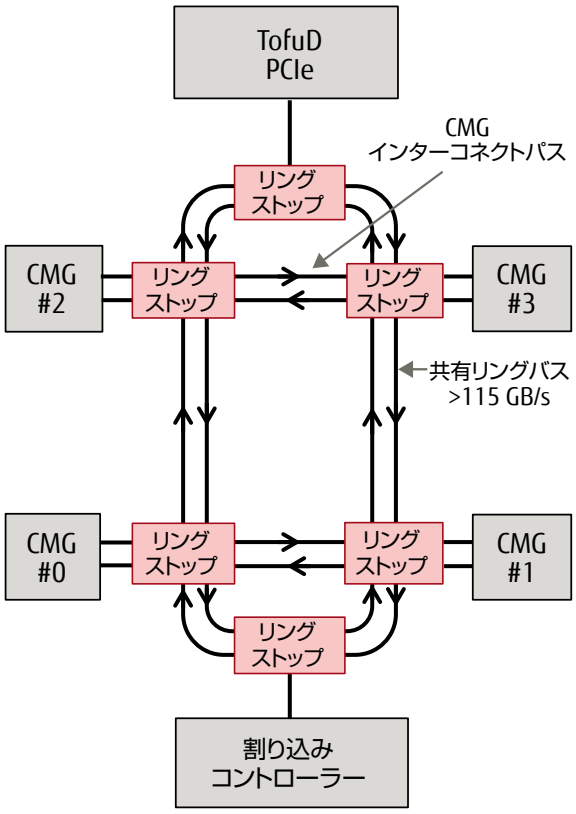

図-5　CMG間の接続図

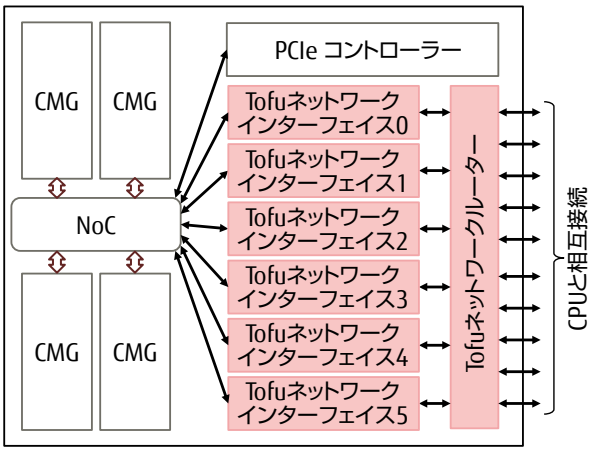

図-6　TofuDのブロック図

4.3 HBM2とメモリーコントローラー

A64FXでは，一般的なサーバで使用されるDDR4 DIMMよりもはるかに高帯域な3次元積層メモリーであるHBM2を採用した。HBM2専用に開発したメモリーコントローラーをプロセッサーに内蔵し，2.5次元パッケージ技術によりCPUチップとHBM2を単一パッケージに異種統合することによって，低レイテンシと1,024 GB/sの高いメモリー帯域を確保した。また，HBM2専用に開発されたメモリーコントローラーは，HBM2メモリーの特性に合わせて性能を最大限引き出すように制御方法の最適化を行うとともに，メインフレームと同等の強固な信頼性を確保した。

4.4 TofuDコントローラーとPCIeコントローラー

A64FXは外部入出力として，CPU同士を相互接続して超並列システムを実現するTofuDと，I/Oデバイスを接続するPCIeバスとを備える。TofuDは伝送速度28 Gbpsの高速シリアル信号を20レーン備え，6.8 GB/sの帯域で最大10個のCPUと相互接続する。PCIeは伝送速度8 Gbpsの高速シリアル信号を16レーン備え，帯域は16 GB/sである。

TofuDのブロック図を図-6に示す。TofuDはネットワークインターフェイス（TNI）を六つ備え，ネットワークルーターを介して10個のCPUと相互接続する。TofuDのネットワークは「京」と同様に，6次元メッシュ/トーラスである[5－7]。ネットワークのノードアドレスは物理的には6次元であるが，ユーザープロセスには仮想的な3次元の座標が与えられ，従来の3次元接続向け通信アルゴリズムを使用できる。TofuDの通信機能も「京」と同様に，ユーザープロセスから直接使用できる機能としてRDMA（Remote Direct Memory Access）通信およびバリア同期通信を，システムがIPパケットの転送に使用する機能としてシステムパケット通信を備える。RDMA通信の種類は「京」と同じPutとGetに加え，「京」の後のTofu2 [8] で拡張されたAtomic Read Modify Writeにも対応する。TofuのRDMA通信は独自の仮想記憶機構を備え，各ノードのOSが管理するユーザープロセスの仮想アドレス空間の間で直接データを転送する。各データ転送にはグローバルプロセスIDが付与され，同じ並列プログラムで実行されるプロセスだけを参照するように保護される。TofuDでは従来の「京」，およびTofu2に比べ，同時通信数，耐故障性，バリア同期通信が強化された。以下にそれぞれの強化された点について説明する。

(1) 約2倍の合計通信帯域

3次元接続に最適化された集団通信アルゴリズムの多くは，3次元6方向に同時通信する。しかし，従来はネットワークインターフェイスが四つであったため，同時に4方向までしか通信できなかった。TofuDでは，ネットワークインターフェイスが六つに増強され，3次元6方向に同時通信して高帯域を実現する通信アルゴリズムを利用できるようになった。同時通信の合計帯域は，「京」の20 GB/sから40.8 GB/sに強化された。

(2) 高い耐故障性

「京」では，隣接ノードへのリンクごとにエラーを検出し，再送信を行っていた。また，エラー検出頻度が高い場合にはリンクを切断した。Tofu2では，リンクを切断した後，半分に縮退したレーン数で自動的に再接続する。既に半分に縮退していた場合には，使用するレーンを変えて再接続したが，レーン数を自動的に回復する手段は実装されなかった。

これに対してTofuDでは，エラー検出頻度が高い場合に帯域を下げるだけでなく，エラー検出頻度が低くなれば下げた帯域を元に戻す機能を実装した。具体的には，エラー検出頻度が高い場合はリンクを維持したまま二つのレーンで同じデータを転送することで耐故障性を高め，エラー検出頻度が低い場合はそれぞれ別のデータを転送する状態に復帰して帯域を回復する。

(3) 六つのバリア同期通信

CMGの導入に対応して，バリア同期通信の資源数を増強し，更に縮約演算の要素数を拡張した。従来は四つのネットワークインターフェイスのうち一つがバリア同期通信を行っていたが，TofuDでは六つのネットワークインターフェイスの全てがバリア同期通信を行う。従来は，データ型に関わらずバリア同期1回当たり1要素を縮約演算できたが，TofuDでは8要素の整数または3要素の浮動小数点数を縮約演算する。

5. 低消費電力設計

低消費電力のために，アーキテクチャーレベルからデバイスレベルまで，広い階層にわたって省電力のための工夫を行った。

5.1 アーキテクチャーレベルの省電力

計算コアをグループ分けして各グループにメモリーを直結する構造にしたことと，グループにまたがるプロセスマッピングを避けることで，アプリケーション動作のほとんどをCMG内にローカライズできるようになった。その結果，データの平均的な移動距離が短縮され，電力を削減している。

5.2 回路レベルの省電力

512 bit SIMDをサポートするに当たり，1次データキャッシュのアクセス方法の見直しを行った。SVEのベクトルロードを多用するアプリケーションの性能はレイテンシよりもスループットが支配的であるため，SVEのロード命令ではレイテンシをわずかに伸ばすことで，1次データキャッシュアクセス時に大幅な電力削減が可能な方式を採用した。

また，演算パイプライン中のデータフォワーディング回路の見直しを行い，消費電力の大きいレジスタファイルの参照を減らして，電力を削減する回路も追加した。

6. むすび

本稿では，A64FXの高性能，高密度実装，低消費電力設計について述べた。

A64FXは幅広いアプリケーションの電力あたり性能を大幅に向上させることを目標として開発した。富士通は理化学研究所 計算科学研究センターとのコデザイン（協調設計）を進めるとともに，プロセッサー開発，システム開発，ソフトウェア開発，コンパイラ開発，性能評価の各チームが緊密に協力し，新技術を開発し，従来技術を発展させ，目標を達成した。

スーパーコンピュータ「富岳」が今後様々な分野の課題解決に貢献するとともに，A64FXが幅広いソフトウェア開発者に受け入れられ，DXを加速させることを期待する。

参考文献・注記

[1] 理化学研究所：Fugaku System Configuration.

https://postk-web.r-ccs.riken.jp/spec.html
[2] T. Yoshida：Fujitsu High Performance CPU for the Post-K Computer, HOT CHIPS 30 (2018). https://www.fujitsu.com/jp/Images/20180821hotchips30.pdf
[3] Y. Ajima et al.：The Tofu Interconnect D. IEEE International Conference on Cluster Computing, p. 646-654 (2018).
[4] S. Yamamura：A64FX High Performance CPU Design, Cool Chips 22 (2019).
[5] 安島雄一郎：スーパーコンピュータ「京」「富岳」を実現した高次元接続技術．富士通テクニカルレビュー（2020）．https://www.fujitsu.com/jp/about/resources/publications/technicalreview/topics/article005.html
[6] Y. Ajima et al.：Tofu: A 6D Mesh/Torus Interconnect for Exascale Computers. IEEE Computer, Vol. 42, No. 11, p. 36-40 (2009).
[7] Y. Ajima et al.：The Tofu Interconnect. IEEE 19th Annual Symposium on High Performance Interconnects, p. 87-94 (2011).
[8] Y. Ajima et al.：The Tofu Interconnect 2. IEEE 22th Annual Symposium on High Performance Interconnects, p. 57-62 (2014).

著者紹介

坂下　聡太（さかした　そうた）
富士通株式会社
プラットフォーム開発本部
プロセッサーの開発に従事。

北村　健一（きたむら　けんいち）
富士通株式会社
プラットフォーム開発本部
プロセッサーの開発に従事。

高木　紀子（たかぎ　のりこ）
富士通株式会社
プラットフォーム開発本部
プロセッサーの開発に従事。

坂田　英樹（さかた　ひでき）
富士通株式会社
プラットフォーム開発本部
プロセッサーの開発に従事。

岡崎　亮平（おかざき　りょうへい）
富士通株式会社
プラットフォーム開発本部
プロセッサーの開発に従事。

石橋　武史（いしばし　たけし）
富士通株式会社
プラットフォーム開発本部
プロセッサーの開発に従事。

田端　猛一（たばた　たけかず）
富士通株式会社
プラットフォーム開発本部
プロセッサーの開発に従事。

中村　武夫（なかむら　たけお）
富士通株式会社
プラットフォーム開発本部
プロセッサーの開発に従事。

安島　雄一郎（あじま　ゆういちろう）
富士通株式会社
プラットフォーム開発本部
アーキテクチャー開発に従事。

この記事は，富士通の技術情報メディア「富士通テクニカルレビュー」に掲載されたものです。他の記事も是非ご覧ください。

富士通テクニカルレビュー

https://www.fujitsu.com/jp/technicalreview/