5H-04

ツインタワーのためのメモリチップ

寺嶋　爽花†　小島　拓也†　奥原　颯†　松下　悠亮†　安藤　尚輝†
並木　美太郎‡　天野　英晴†
†慶應義塾大学 ‡東京農工大学

1　はじめに

ビルディングブロック型計算システムは、様々な機能のチップを組み合わせて大規模なシステムを構築することで、低コストかつ用途に応じて柔軟にシステムを構成することを目的としている。このようなシステムを構築するためには、スタックされた全てのチップからアクセス可能な大規模なメモリが重要である。また、複数のチップを積層する場合、単純な三次元積層では電源の配線や、チップをずらして積層することによる物理的な安定性の観点から限界が生じる。そこで、大規模なDRAM を TCI(Thru-Chip Interface)[1] を用いて、複数のチップ積層（ここではタワーと呼ぶ）と組み合わせることが可能なシステムが必要になる。しかしながら、今回は DRAM による共有メモリは実現できなかった。ここでは、このシステムのプロトタイプとして、SRAM による共有メモリチップ上に二つのタワーを積層することのできるチップ SMTT(Shared Memory for Twin Tower) を開発した。本稿では SMTT によるツインタワーアーキテクチャと評価について述べる。

2　アーキテクチャ

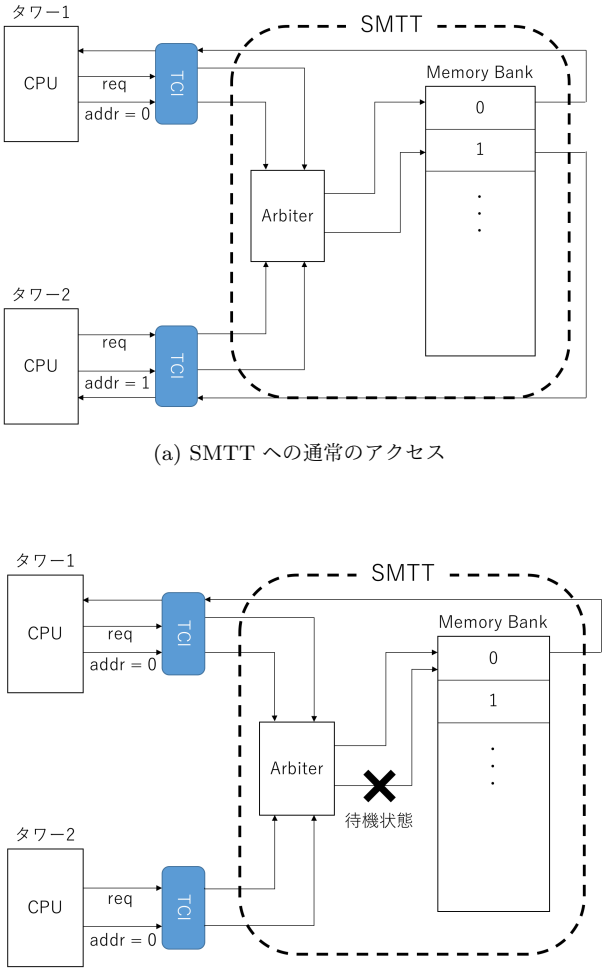
2.1　SMTT へのアクセス

図 1 は SMTT のアーキテクチャの概要図である。各タワーのホスト CPU から TCI を介して要求信号を送る。この信号はデータやアドレス、Read/Write などの命令が組み込まれており、Arbiter がこれをデコードしてメモリアクセスを行う。SMTT のメモリは複数のバンクに分かれており、図 1a のように 2CPU から異なるメモリバンクへ並列にアクセスすることが可能であるが、2CPU から同時に同一のバンクにアクセスすることはできない。この場合は、図 1b のように Arbiter が片方のメモリアクセスを待機させ、もう一方の CPU からのメモリアクセスを完了させた後に、待機させていた CPU のメモリアクセスを再開させる。

ツインタワーシステムは、SMTT を土台としてその上にチップを積んでいくことで形成する。各チップおよび SMTT は TCI を用いて接続し、各タワーの一番上にあるチップをホスト CPU とする。ホスト CPU からタワー内の各チップへのアクセスは、ホスト CPU のアドレス空間に各チップの領域をマッピングし、これらの領域に Load/Store を行うことで実現する。

2.2　同期レジスタ

SMTT は同期操作を実現するために同期レジスタを備えている。同期レジスタは、Fetch&Dec 命令が実装されており、不可分に処理することができる。例えば、ある共有変数をセマフォとして用いるとする。双方のタワーがこの変数を参照する際に、片方のタワーが書き込みを行う前にもう一方のタワーから読み込みが実行されてしまうということが考えられる。このような状態を防ぐために同期レジスタを用いる。同期レジスタの書き込みは通常のレジスタへの書き込みと同様に行われる。読み込み時は、同期レジスタはその時保持している値を出力すると同時にデクリメントされた値が書き込まれる。ただし、保持している値が 0 の場合はデクリメントしない。



(a) SMTT への通常のアクセス

(b) アクセスが競合した場合

図 1: SMTT のアーキテクチャ

3　チップ実装

図 2 に Silicon on Thin Box (SOTB) 65-nm プロセス [2] で実装した SMTT のレイアウト図を、表 1 に実装仕様を示す。SMTT のバンクのそれぞれは 32KByte のシングルポート SRAM の IP を用いて実装しており、8 つのメモリバンクで計 256KB の容量を持つ。チップサイズは 6mm×6mm で、標準の電源電圧は 0.75V である。右上と左下には TCI の IP が配置されており、タ

表 1: SMTT の実装仕様

SMTT	メモリ 同期レジスタ	SRAM 32KB× 8 32 bits × 32
Chip	プロセス サイズ	SOTB 65nm 7-metal 6mm × 6mm
Tools	設計 論理合成 配置配線	Verilog HDL Synopsys Design Compiler 2016.03-SP4 Synopsys IC Compiler 2016.03-SP4

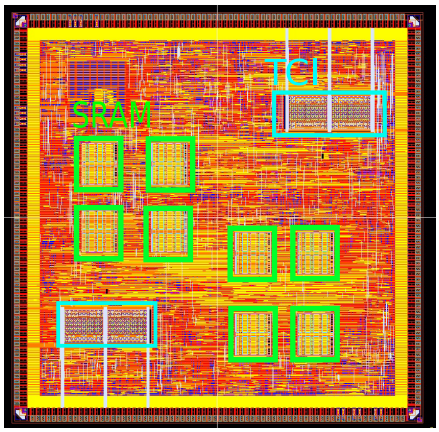


図 2: SMTT のチップレイアウト

ワーはこの上に構築することが可能である。SRAM と TCI の IP 以外のコントローラや同期レジスタなどはチップ全体にちりばめるように配置配線している。

4 評価

シミュレーションに用いた環境を表 2 にまとめる。配置配線により得られたレイアウトを用いて IC-Compiler から最大動作周波数、消費電力を求めた。その結果を表 3 に示す。

64 点高速フーリエ変換 (FFT) プログラムを用いて実行時間の検証と比較を行った。本稿で比較する対象は、SMTT を用いない GeyserSCM[3] 1 コアと SMTT を用いたツインタワー (GeyserSCM2 コア) である。検証に用いた FFT のプログラムは、入力データはプログラム上で用意し、GeyserSCM で処理を行う。このとき、ツインタワーでは各タワーで並列にバタフライ演算を行い、中間データは SMTT を用いてやり取りを行う。その後、算出結果を GeyserSCM のメモリに格納する。それぞれの実行結果を図 3 に示す。ツインタワーは、SMTT を通して中間データの送受信を行うため、同期と転送を行う必要があり、全体に対するオーバーヘッドは約 4.5%である。実行にかかる合計サイクル数は、シングルタワーでは 28179 サイクル、ツインタワーでは 19777 サイクルであった。結果として、ツインタワーの実行時間は、シングルタワーに比べて 42.5%の高速化を達成した。

表 2: 使用した評価ツール

動作シミュレーション	Cadence NC-Verilog 10.20-s131
電力シミュレーション	Synopsys Prime Time 2012.12-SP3

表 3: SMTT の性能評価

最大動作周波数	53.79MHz
ダイナミック電力	11.94mW
リーク電流	12.31μW

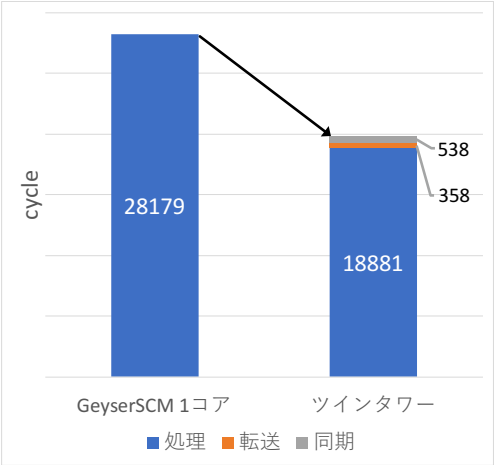


図 3: FFT の実行速度

5 結論

ビルディングブロック型計算システムのための共有メモリチップのプロトタイプとして，SRAM で構成された共有メモリ SMTT の開発と評価を行なった。これにより、単純な三次元方向のチップの積層から、ブリッジとして SMTT を導入することで、2 倍のチップ数の積層を可能にした。また、パフォーマンスについては 64 点高速フーリエ変換を行った際の SMTT の有無による実行時間の比較を行った。ツインタワーにおける同期と転送時間の全体に対するオーバーヘッドは約 4.5%であった。シングルコアの場合と比較すると、およそ 43%の高速化がなされており、性能を向上させることができた。

参考文献

[1] Y. Take, H. Matsutani, D. Sasaki, M. Koibuch, T. Kuroda, and H. Amano, “3-D NoC with Inductive-Coupling Links for Building-Block SiPs,” IEEE Transactions on Computers(TC), vol.63, no.3, pp.748–763, March 2014.

[2] Y. Morita, et al., “Smallest Vth variability achieved by intrinsic silicon on thin BOX (SOTB) CMOS with single metal gate,” 2008 Symposium on VLSI Technology, pp.166–167, June 2008.

[3] Y.Yoshida and K.Usami, “Energy-efficient standard cell memory with optimized body-bias separation in silicon-on-thin-box (sotb),” IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol.E100.A, no.12, pp.2785–2796, 2017.