C-018

ユビキタスプロセッサの最適設計

Optimal Design of a Ubiquitous Processor

成田 一貴† 内海 晴信† 石原 拓美† 三村 直道† 高木 竜哉†
Kazuki Narita Harunobu Uchiumi Takumi Ishihara Naomichi Mimura Tatsuya Takaki
深瀬 政秋† 佐藤 友暁‡
Masa-aki Fukase Tomoaki sato
†弘前大学大学院理工学研究科
‡弘前大学総合情報処理センター

1. まえがき

IT 革命をもたらしたユビキタス化の本質は、大小各種の膨大な数のマルチメディア情報機器群を世界規模で分散し、ネットワーク結合させることにあるため、情報流出、情報洪水、ディジタル格差等の課題を必然的に内包する。周知の通り、その常套的な対策は組み込みソフトの強化であるが、このやり方は必然的にメモリ増加を伴う。従って、本来簡易であるべきプラットフォームの負担を増し、モバイル化に逆行しているはずである。一方、ポストユビキタスに向けてディペンダブルコンピューティングやシンビオティックシステムなどが研究されつつある。これらは総合的な分野を包含し、その理解と取扱いには高度な IT 知識が前提となりがちなので、一般的な普及は必ずしも容易ではない。また、依存するソフトが大がかりである故に、大量マルチメディアデータの迅速処理には向いていない。

最近の新たな社会的要求として、きめ細かな環境対策、災害時の機能性と迅速対応、移動型サービスによる地域社会の質的向上などが加わった。このため、恒久的なネットワーク基盤とは独立に簡易敷設するアドホックネットワークの重要性が増している。しかし、アドホックネットワークでは、コストパフォーマンスや簡易性とセキュリティ脆弱性との二律背反が一層顕著になる。

我々は以上のようなユビキタス化にまつわる諸課題に対してハード的アプローチをとり、ユビキタスプロセッサ HCgorilla の開発を行ってきた [1]。HCgorilla はダブルコア構造で、各コアは Java 対応型のメディア処理用パイプラインとマルチメディアデータに対する過渡的なセキュリティを確保するためのサイファーパイプラインを備える。このようにモバイルと環境に対する省電力性、マルチメディアに対する高速高機能性、ユーザのプログラミングに対する柔軟性、ユビキタスに対する信頼性の全てを追求し、上述の諸課題をクリアすることを目的とすることから、HCgorilla をユビキタスプロセッサと呼んでいる。

ユビキタス化に必要な省電力化と高速化について、HCgorilla にはまだ開発の余地があることから、本論文では HCgorilla の総合評価を行い、現時点での課題を明らかにする。さらに、今後の最適設計指針について述べる。

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、シノプシス社の協力で行われたものである。

2. HCgorilla の概要

表 1 に HCgorilla の開発履歴をまとめる。HCgorilla は 2 つの独立したコアを持ち、各コアはメディアパイプとサイファーパイプで構成される。これらのパイプラインに応じてデータキャッシュが独立している。HCgorilla は Java 対応で PC プロセッサ並みの柔軟性を持たせるために、HCgorilla.4 で FPU (floating point unit)を搭載した。マルチメディア処理の計算アルゴリズムでは浮動小数点が多用されるが、組込みプロセッサではチップの制約から浮動小数点演算を固定小数点化することが多い。HCgorilla.4 は浮動小数点演算命令を有している。しかし、FPU のレイテンシィは 5 クロックで、2 クロックの IU (integer unit)との間でスケジューリングを必要とした。この課題を解決するため、HCgorilla.4 の実行段をウェーブ化 MFU で置き換えたものが HCgorilla.5 である。

本研究では、図 1 に示す HCgorilla.6 について評価する。HCgorilla.6 の特徴は以下のとおりである。メディアパイプについては、

・6 段で、200MHz で動作。
・16bit×64word のインストラクションキャッシュと、16bit×8word のスタックを 4 個搭載。
・231 個の Java バイトコードのうち 58 個を直接実行。102 個は、別途開発中のソフトウェア支援で展開。

サイファーパイプは、乱数発生器 (random number generator, RNG)を内蔵する。レジスタファイルからデータキャッシュに転送する間に暗号、復号用の SIMD 命令を実行することで、暗号ストリーミングを行う。

HCgorilla の開発と本研究の評価の環境を表 2 にまとめる。ワークステーションは IBM 社の A Pro OPT254 W/NVS285 RHL を使用する。主な評価は Synopsys 社の Design Compiler で行い、厳密な場合には Synopsys 社の primetime を用いる。

表 1　HCgorilla の開発履歴

Version				HCgorilla.3	HCgorilla.4	HCgorilla.5	HCgorilla.6
Chip parameter	Design Rule			ROHM 0.18μm CMOS			
	Area		Chip	5.0mm × 7.5mm			
			Core	4.28mm × 6.94mm			
	Power Supply			1.8V (I/O 3.3V)			
Per processor	Power consumption			241mW		274mW	
	Clock frequency			330MHz	400MHz	200MHz	
	Instruction cache			16bit × 32word × 2		16bit × 64word × 2	
	Data cache			16bit × 128word		16bit × 128word × 2	
	Stack memory			16bit × 8word × 4		16bit × 16word × 8	
	Register file			16bit × 64word		16bit × 128word	
	No. of cores			2			
Pipelines/ core	Arithmetic	No.		2/core			
		Per arithmetic pipe	EX IU	2-wave		2-wave MFU	
			EX FPU	NA	5-clock		
			Stacks	1		2	
		ILP degree		2		4	
		Throughput			0.07-0.09 GIPS	0.16 GIPS	
		Java bytecode		61	77	58 (102)	
	Cipher	No.		1/core			
		RNG		4-bit LFSR × 1		6-bit LFSR × 1	6-bit LFSR × 2
		Throughput		0.1-0.2 GOPS			
		SIMD cipher code		2			

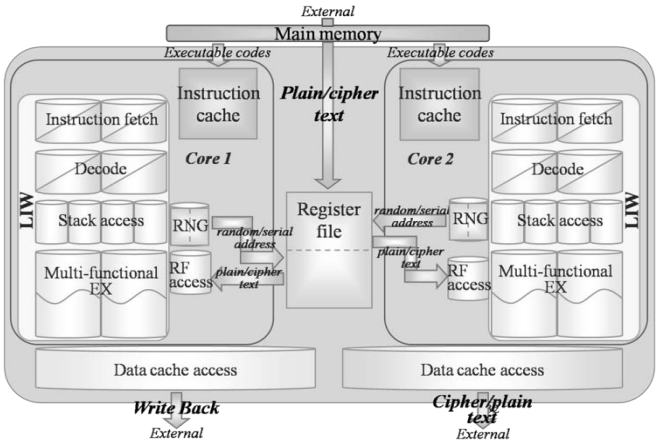


図 1　HCgorilla の基本構成

表 2　開発環境

Software	
OS	Red Hat Linux 4/SentOS 5.4
Synthesis tool	Synopsys - Design Compiler D-2010.03
Simulation tool	Synopsys - VCS version Y-2006.06-SP1
Physical Implementation tool	Synopsys - IC Compiler C-2009.06
verification tool	Mentor - Calibre v2010.02_13.12
Equivalent verification tool	Synopsys – Formality B-2008.09-SP5
Static Timing analysis tool	Synopsys – Primetime pts,vA-2007.12-SP3
Simulation tool	Synopsys - hspice simif D-2010.03 or hsim_vD-2010.03
Language	
Synthesis	VHDL
Simulation	Verilog - HDL
Technology	
Kyotouniv ROHM 0.18μm CMOS Standard Cell Library	

3. HCgorilla.6 の評価

HCgorilla.6 のメディアパイプとサイファーパイプについてスループット及び消費電力、面積の評価を行う。

3.1 Media pipe

メディアパイプの評価には、表 3 に示すテストプログラムを使用する。ASV (Advanced Safety Vehicle)や ITS (Intelligent Transport Systems)分野で使われる正規化相関の一部の処理を再現するため、演算数とループカウント変数に整数型、フロート型を組み合わせたプログラムルーチン A,B,C を用意した。

表 3　テストプログラム

Pipeline	Process
Media pipe	$\sum_{k=1}^{x} k$ ， $x = 2^n$, n=1 to 8 Routine / Variable k / Loop count Routine A / Integer / Integer Routine B / Float / Float Routine C / Float / Integer
Cipher pipe	Double cipher of image data Register file length 32,64,128,256,512 word 1word=2byte

図 2 にメディアパイプのスループットを示す。Routine A, B, C 全ての処理において 0.17GIPS となった。整数型、浮動小数点型が混在することによる命令スケジューリングの複雑化が解消されていることを示せた。

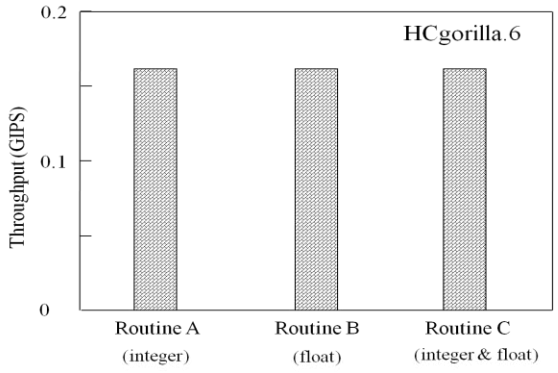


図 2　Media pipe’s throughput

図 3 にルーチン A を実行した場合の HCgorilla.6 全体の消費電力を示す。電力成分の内訳を以下に示す。

Dynamic power
= Cell internal power + Net switching power　(1)
Cell internal power
=各セルの内部で消費される電力
= short circuit power + switching power　(2)
Net Switching power
=回路内ネットの負荷容量による消費電力　(3)

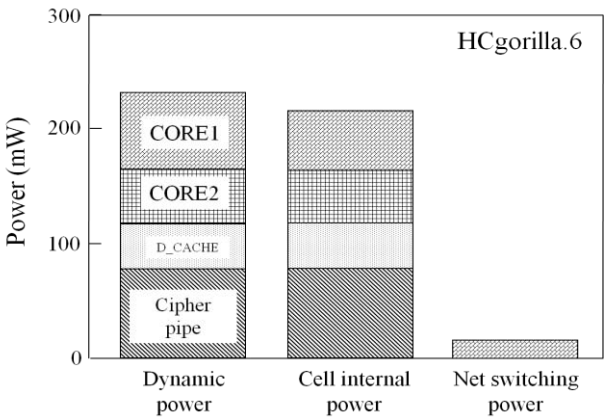


図 3　HCgorilla.6 の消費電力

図 3 より、全体の消費電力 232mW に対して、プログラムを実行しているメディアパイプの Core1 は 67mW である。一方、計算結果の書き込み時のみ利用されるデータキャッシュ、何も処理を行っていない Core2、サイファーパイプ、つまりスイッチングがあまり行われていない部分でも合わせて 164mW 消費していた。この電力の成分は Core1 の Net switching power を考えた場合そのほとんどが short circuit power であると推測される。このことからスイッチングが行われていない部分のクロックを停止することで、消費電力の削減に繋がる。

3.2 Cipher pipe

サイファーパイプの評価は QVGA サイズの画像を暗号化することで行う。面積、消費電力とスループットのトレードオフを探るため、Core 当たりのレジスタファイル、データキャッシュをそれぞれ 64word、128word、256word、512word、1024word とした回路を用意した。サイファーパイプの電力評価はスイッチング情報を加味せず、論理合成ソフトから得られる目安値を用いる。

図 4 はサイファーパイプのスループットをレジスタファイル、データキャッシュの容量ごとに示したものである。32word で 0.18GOPS、512word で 0.19GOPS であり、128word 以降のスループットの向上は少なかった。図 5 はレジスタファイル、データキャッシュの容量ごとの HCgorilla.6 全体の消費電力と面積である。電力、面積ともに 128word から

大幅に増加していることが分かる。図 4, 5 から、レジスタファイル、データキャッシュの容量は 128word 以下が望ましい。

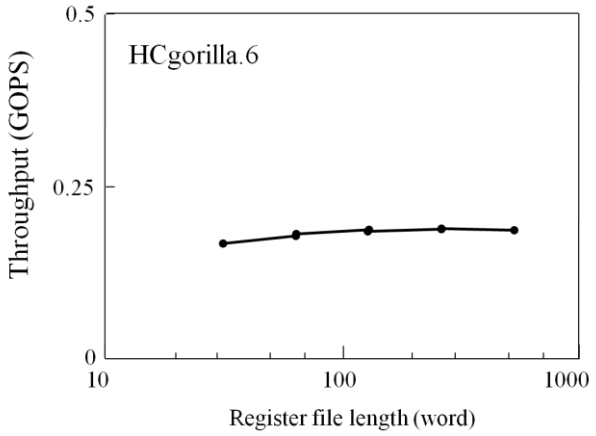


図 4　Ciper pipe's throughput

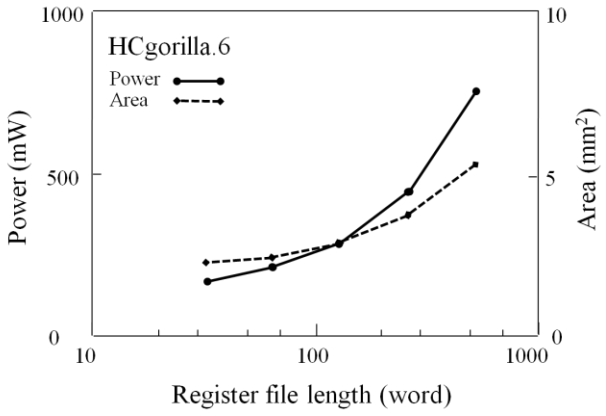


図 5　Power & Area

4. デスカッション

0.18μm プロセスの CMOS チップに実装した HCgorill では、ダイナミック電力の占める割合が極めて大きい。従って、チップの省電力化のためにはこの電力成分の削減がポイントである。そのための指針を次式の静的評価で考えてみる。

$$P=\sum_{\text{all cell}} P_{cell} = \sum_{\text{the clocked cells}} P_{cell} \quad (4)$$

$$P_{cell}=A\{CV^2f+\tau VI_{short}f\}+VI_{leak} \quad (5)$$

$$f_{max}= (V- V_{th})^2/V \quad (6)$$

ここで、

P: プロセッサ全体の消費電力

P_{cell}: 1 セルあたりの消費電力

A: クロック供給中の実動セルが 1 クロックサイクル内にスイッチングする確率

$C=C_{cell}+C_{load}$

C_{cell}:セル内の浮遊容量（印加電圧は第 1 近似）

C_{load}: セル出力側の負荷容量

V: 電源電圧

V_{th}: MOS のスレッショールド電圧

f: クロック周波数

f_{max}: クロック周波数の最大値

τ: オンオフ切り替え時に電源から実動セルを介してアースに短絡電流が流れる時間

I_{short}: 短絡電流が流れている全期間の平均値

I_{leak}: リーク電流

である [2]。

式(5)より、ショートサーキット電力削減のために例えば V を減らせば f が減り、f を減らせばスループットが劣化する。このように、セルレベルの省電力化には複雑なトレードオフを伴うので、抜本的な成果を得難い。そこで、式(4)に基づいて、マイクロアーキテクチャレベルの省電力化策を考える。

マイクロアーキテクチャレベルの省電力化の常套手段はゲーテッドクロックであるが、この確立した技術を HCgorilla に直接導入することはできない。何故なら、HCgorilla はウェーブ化がなされているためである。CAD ツールは通常型パイプラインの設計に特化されていることから、HCgorilla の実行段のウェーブ化はマニアルチューニングの対応がなされてきた。CAD ツールがサポートするゲーテッドクロックは、HCgorilla には適用できない。

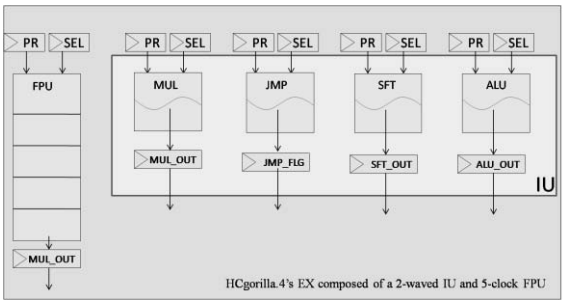
4.1 ウェーブ化 MFU

図 6 に HCgorilla の実行段を示す。図 6(a)に示した base MFU を、ラフチューニングにより図 6(b)のようにウェーブ化する。図 7 に waved MFU のクロックスピードと消費電力を示す。横軸はウェーブ化の次数で、従来型パイプラインの段数に匹敵する。Waved MFU は、base MFU の命令スケジューリング（整数演算と浮動小数点演算の間で必要）の課題を解消するが、ウェーブ化自体の本来の特徴は高速省電力の両立にある [3]。しかし、図 4.1.2 ではその特徴が発揮されていない。そこで、今後は

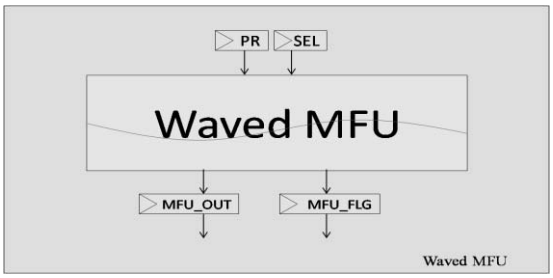
・レイアウト設計とチューニングを連携させる。

・ファインチューニングを行うことで、ばらつきに配慮したウェーブ化を行う。

以上に着手する。



(a)



(b)

図 6　(a) Base MFU. (b) Waved MFU.

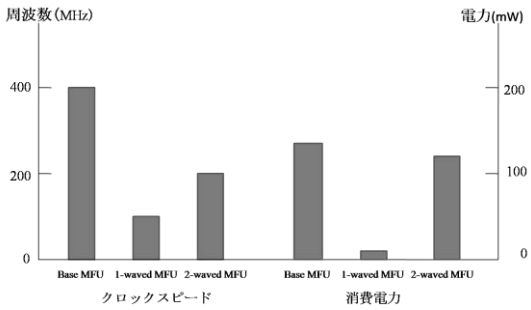


図 7　クロックスピード＆消費電力

4.2 ゲーテッドクロック

ウェーブ化を施した HCgorilla には CAD ツールがサポートするゲーテッドクロックを適用できないことから、ここではゲーテッドクロックのマニアル手法を独自に吟味する。データパス、制御系、クロックや通常パイプラインとウェーブパイプラインの完全な融合はありえないので、マニアル手法は妥当で意味のあるアプローチである。更に、HCgorilla にテスト容易化の機構を加える。ゲーテッドクロックと scan logic はどちらもパイプラインレジスタに対する細工なので、各々のクロックスキームを融合する必要がある。

図 8 に HCgorilla のメディアパイプに対するクロックスキームの融合を示す。ゲーテッドクロックのマニアル手法について記述せよ。Scan logic は各パイプラインレジスタに対して値の確認、変更を可能にすることで動作検証容易にする。IFU (Instruction Fetch Unit)、IDU (Instruction Decode Unit)、SAU (Stack Access Unit)、MFU のレジスタ部をシフトレジスタで構成し、scan control で任意のユニットに切り替え、シリアル入出力 scan in、scan out でレジスタの値を確認、変更を行う。Clocking gate は IDU の制御信号を基に、動作すべき回路にのみクロックを供給し、消費電力を低減する。

図 9 と 10 に Gated clock 組み込んだメディアパイプのシミュレーションを示す。図の通りに、4.3.2.1 の POP 命令と 1 と 2 の加算命令について、説明する。命令読み込中では、すべてのクロックが立ち上がる。SAU 部分(DC_W_ADRS_L,DC_W_ADRS_U,DC_W_DATA_L,DC_W_DATA_U)と MFU 部分にデータが来てない状態では Clocking gate(N_GCLOCK_MFU_L, N_GCLOCK_MFU_U, N_GCLOCK_SAU_DSEL_L,N_GCLOCK_SAU_DSEL_U, N_GCLOCK_SAU_STK_L, N_GCLOCK_SAU_STK_U)が立ち下がったため、SAU 部分と MFU 部分が停止する。
図 11 に Gated clock に応じて SCSEL の修正した後のシミュレーションを示す。図の通り、SAU_L(DC_W_DATA_L)の部分と SAU_U(DC_W_DATA_U)の部分の SCSEL 命令に対して、それぞれ正しく動作しているのと SCSEL が正常に動作しているのが確認できる。

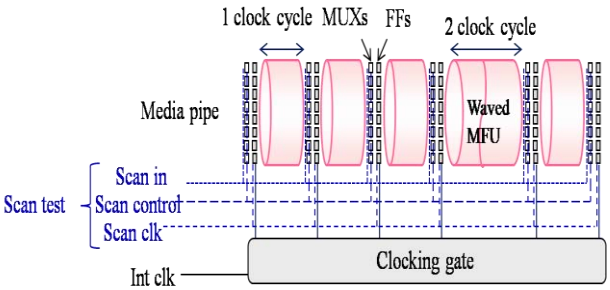


図 8　クロックスキームの融合

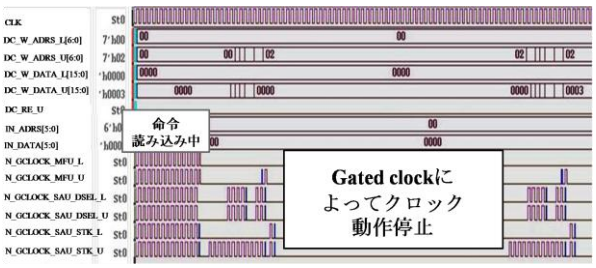


図 9　Gated clock シミュレーション

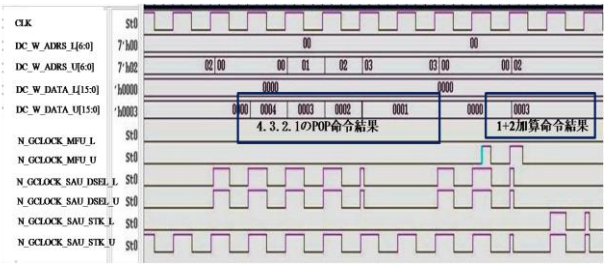


図 10　Gated clock シミュレーション拡大図

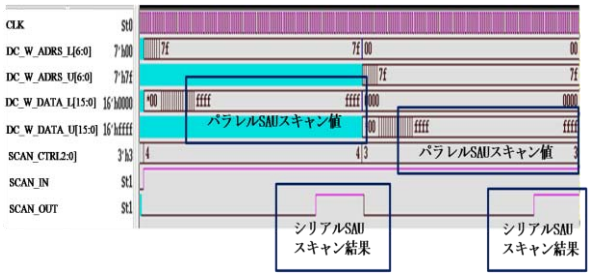


図 11　Clocking gate 用 SCSEL

図 12 と図 13 にメディアパイプの面積と消費電力を示す。消費電力は静的な評価に基づくので厳密ではないが、ゲーテッドクロックは面積のオーバヘッドを伴わずに消費電力を削減できることが確認できる。

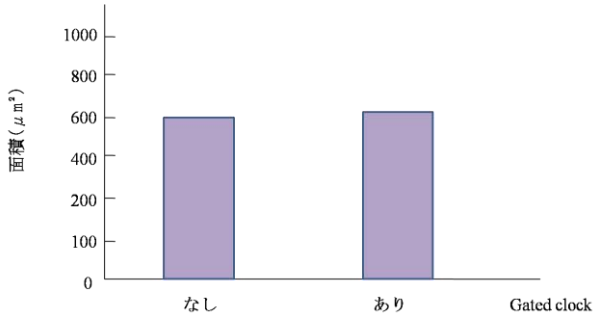


図 12　メディアパイプ面積

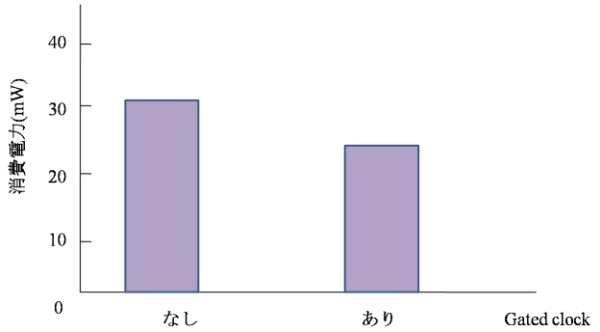


図 13　メディアパイプ消費電力

5. むすび

本論文は、ユビキタスプロセッサ HCgorilla6 の設計と総合評価について述べた。メディアパイプの評価より、稼働率が低くて無駄にスイッチングが行われている部分のクロックを停止すること、即ちゲーテッドクロックの必要性を確認した。ゲーテッドクロックとウェーブパイプラインのクロックスキームの融合についても述べた。

今後の課題は、Gated clock の導入効果の動的評価である。また、最適設計に向けてウェーブ化におけるより詳細なパス遅延解析手法及び専用バッファの開発と、融合型クロックスキームの HCgorilla チップへの導入である。

文　献

[1] 内海晴信、石原拓美、三村直道、成田一貴、高木竜哉、深瀬政秋、佐藤友暁「ユビキタスプロセッサのチップ開発」信学技報，Vol. 110, No. 344, pp. 43-48 (ICD-2010-102), 2010.

[2] T. Mudge, “Power: A First-Class Architectural Design Constraint,” Computer Magazine, Vol. 34, No. 4, pp. 52-58, April, 2001.

[3] M. Fukase and T. Sato, “Design Techniques of Wave Pipelines,” IPSJ SIG Notes, Vol.2007, No.55, pp. 67-72 (2007-ARC-173), 31 May 2007.