2007－DPS－131 (16)
2007／6／7

HCgorillaのマルチメディア機能強化

○ 野田一訓[1] 武田宏樹[1] 深瀬政秋[1] 佐藤友暁[2]
1 弘前大学大学院 理工学研究科 電子情報システム工学専攻 電子情報機器学講座
2 弘前大学総合情報処理センター

概要

本研究では,我々が開発してきた,ユビキタスネットワークに貢献し得る,マルチメディアモバイルプロセッサgorillaと大容量データの暗号処理を行う事が出来るストリーム暗号エンジンRAC(Random Addressing Cryptography)との統合を行った,ハードウェア暗号組込み型マルチメディアモバイルプロセッサHCgorillaのマルチメディア機能強化を目的として,浮動小数点数演算ユニット(Floating point number Processing Unit : FPU)を開発する.また,HCgorillaアーキテクチャの命令セット拡大・Decode stageの改良を行い,HCgorilla.4としてFPUの実装を行う.

A multimedia performance enhancement of HCgorilla

○ Kazunori Noda[1] Hirolki Takeda[1] Masaaki Fukase[1] Tomoaki Sato[2]
1 Department of Electronic and Information Systems Engineering HIROSAKI Univ.
2 Computer and Network Systems Center HIROSAKI Univ.

Abstract

Progressive ubiquitous networks have impressed us with alternative features, diversity or security. When the diversity from small devices to large machines is in normal states, ubiquitous networks are functional and useful. But, it is hard to control if abnormal states once happen. Since the diversity brings about open-access to ubiquitous networks anytime anywhere, this really threatens user security. The key to protect huge amount of multimedia data in ubiquitous networks is to introduce safety aware high-performed single VLSI processor systems embedded with cipher process. Thus, we have exploited the architecture of a hardware cryptography-embedded multimedia mobile processor named HCgorilla by sophisticatedly unifying up-to-date processor techniques. In this research, we further enhance HCgorilla's Java functions. Aiming at a multimedia performance enhancement, this paper focuses on FPU (Floating point number Processing Unit). By the improvement of Decode stage and the instruction set enlargement of HCgorilla, FPU is mounted on HCgorilla.4.

1. はじめに

携帯機器の分野では,機能充実のためにJVM(Java Virtual Machine)が主流になっている.しかし,導入が進められてきたのはJIT型のJavaチップで,ソフトウェアのJVMを内蔵する大容量キャッシュメモリの電力消費を削減するために1トランジスタ化のモバイルRAMや,バッテリの改良が強いられている.当研究室では,こういった問題の根本的な省電力化対策としてJVM cacheを使わず,マイクロプラグラム方式でJavaバイトコードを直接実行する,インタープリタ型Javaチップを開発してきた.

また,近年,ワイヤレスネットワークの急速な発展に伴い,ユビキタスネットワーク時代が到来している.モバイルPCや携帯電話,PDAなどといったもの以外にも,ICタグといった非常に小さなものまでユビキタスネットワーク基盤の要素となっている.

ユビキタスネットワーク社会では至るところでネットワークが形成されている分,第三者による攻撃や侵入,また,盗聴や情報漏えいの危険性も大きくなる.利便性や機能性は増すが,多様的に広がったデバイスは予想外の攻撃が一度起きたときの脆弱性は大きい.ユビキタスネットワークの実用的な安全対策として,Firewallや IDS(Intrusion Detection System)・IPS(Intrusion Protection System)などが一般的に使用されている.しかし,ほとんどの場合これらはソフトウェアであり,処理の複雑さとコンピュータのパフォーマンスはトレードオフの関係にある.またユビキタスネットワークで用いられるデータは,実際ほとんどの場合,大容量のマルチメディアデータであるので,低消費かつ高速な暗号化という事が重要になってくる.

このような背景から,我々はユビキタスネットワーク社会に貢献しうるマルチメディアモバイルプロセッサの開発や,ハードウェアで暗号

処理を行うプロセッサの開発を行ってきた. HCgorilla は, Java CPU であるマルチメディアモバイルプロセッサ gorilla とストリーム暗号エンジン RAC を併合したハードウェア暗号組み込み型マルチメディアモバイルプロセッサである.

2. HCgorilla

本章ではアーキテクチャ HCgorilla.3 の特徴を示す.

2.1. Java CPU

HCgorilla.3 は Java CPU の実行形式を取っており, Java instruction を高速実行する事が可能となっている. Java 61 命令に, RAC の暗号化・復号化 2 命令を加えた 63 個の命令セットを持つ.

2.2. Multi-core

Multi-core は, CPU ダイの上にそれぞれ単独で処理が可能な CPU を複数実装することで, 性能を引き出す為のプログラミングが必要ではあるが, CPU チップ全体としての処理能力の向上に貢献する. HCgorilla.3 の場合は, 1 つの CPU ダイの上に 2 つの CPU コアを実装している. また, Java 自体がマルチスレッドに対応している為, プログラミングの時点で並列化を意識すれば非常に効率的に処理を実行する事が出来る.

2.3. LIW

LIW は依存関係のない多数のインストラクションを一塊として同時並列処理を実行する技術である. HCgorilla.3 の場合は, CPU コア内に配した独立実行段の数だけの命令を一塊として同時並列処理を実行する事で LIW を実現している. HCgorilla.3 の場合, Multi-core と LIW の実装によって最大 4 並列処理を実行する事が出来る.

2.4. RAC

RAC (Random Addressing Cryptography) の基本原理は, RNG (Random Number Generator) によって生成された擬似乱数を用いたランダムアドレッシングによる転置暗号方式である.

RNG の初期値は公開鍵方式で受信側に送信される. RNG の擬似乱数を同期させ暗号化や復号化を行う事は共通鍵方式である.

暗号化の場合, RNG から出力した乱数と Data cache のアドレスを対応させ, 転置暗号を行う. 例えば, RNG の出力として "1 3 4 0 2" という擬似乱数が生成されたとする. その場合, 擬似乱数の '1' と Data cache のアドレスを対応させ, Register file の 0 番地のデータを Data cache の 1 番地に書き込む. 続けて次の擬似乱数 '3' と Data cache のアドレスを対応させ, Register file の 1 番地のデータを Data cache の 3 番地に書き込む. 以降, 同様に転置暗号化を行う.

復号化の際は, RNG から出力した乱数と Register file のアドレスを対応させ, 転置復号を行う. 当然, 入力するデータは同じ初期値で RAC により暗号化されたデータを用いる. RNG の出力として暗号化の際と同様の "1 3 4 0 2" という擬似乱数を使用し, 擬似乱数の '1' と Register file のアドレスを対応させ, Register file の 1 番地のデータを Data cache の 0 番地に書き込む. 続けて次の擬似乱数 '3' と Register file のアドレスを対応させ, Register file の 3 番地のデータを Data cache の 1 番地に書き込む. 順次, 同様に転置復号化を行う. Figure 1 に RAC の概要を示す.

大容量データの暗号化には高速に処理することが出来る共通鍵方式が非常に有効であり, 大容量データの通信が盛んに行われるユビキタスネットワークの様々な要求に答えることが出来る.

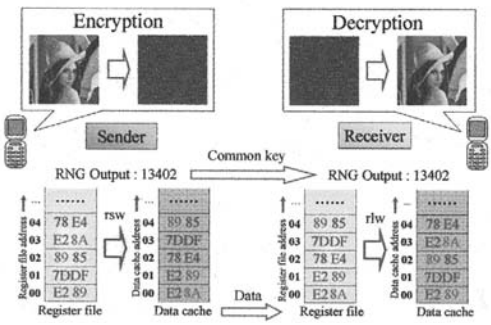


Figure 1. RAC 概要

2.4. Pipeline

HCgorilla.3 は 7 段パイプライン構成であり, Execute / Register file access stage に 2 段の wave-pipeline 化を施している. また, Instruction cache への Executable code 書き込みや Data cache から外部への読み出しはパイプライン外部で行われる.

3. 研究目的

本研究ではこの HCgorilla の Java 機能に焦点を当て, マルチメディア機能強化を目的として, HCgorilla に実装するための浮動小数点数演算ユニットの開発・実装を行う. これは, 近年, 携帯電話や PDA をはじめとするモバイル機器にも 3D グラフィックス処理や, 写真の画像処理機能等が内蔵されてきていること. また, JVM にも float

型・double型命令が実装されていることに起因し，通常の演算の他に浮動小数点数演算のハードウェアレベルでの実行が必要とされているためである．そこで，当研究室でこれまでに開発されたFPUに新たな機能として「乗算」「例外処理」「漸近的アンダーフロー」の回路を組込む．また，遅延制約によりHCgorillaと同じ400MHzでの動作，パイプライン制御の最適化を行い，HCgorillaに十分実装可能なFPUを開発し，HCgorillaの命令セット拡大・Decode stageの改良と併せて新たにHCgorilla.4としてFPUを実装する．

4. 浮動小数点数

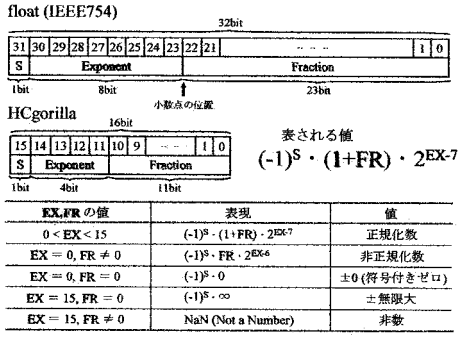
4.1. IEEE754

IEEE 754 (IEEE 二進化浮動小数点数演算標準)とは，浮動小数点数の計算で最も広く採用されている標準規格であり，JVMにも採用されている．この標準規格は，±0や非正規化数等の浮動小数点数の表現形式を規定するとともに，無限大やNaNといった特殊な値も規定し，浮動小数点数演算でのそれらの値の扱い方を規定している．また，4種類の丸めモードと，例外処理も規定している．

4.2. FPU.2について

JVMの浮動小数点数表現には単精度 (float) 型32bit，倍精度 (double) 型64bitが採用されている．これまでに当研究室で開発されたFPU.2はハードウェア暗号組込み型マルチメディアモバイルプロセッサHCgorillaへの実装のために，入出力のビット幅を一致させるために16bitのfloat型とした．また，7段パイプライン構成で最も遅延時間のかかる実行段をWave化することで315MHzでの加減算の動作を可能とする．

Figure 2.にIEEE754とHCgorillaのfloat型についての詳細を示す．



EX,FRの値	表現	値
0 < EX < 15	$(-1)^S \cdot (1+FR) \cdot 2^{EX-7}$	正規化数
EX = 0, FR ≠ 0	$(-1)^S \cdot FR \cdot 2^{EX-6}$	非正規化数
EX = 0, FR = 0	$(-1)^S \cdot 0$	±0 (符号付きゼロ)
EX = 15, FR = 0	$(-1)^S \cdot \infty$	±無限大
EX = 15, FR ≠ 0	NaN (Not a Number)	非数

Figure 2. float型 詳細

5. FPU Architectureの最構築

まず，新しいFPUアーキテクチャFPU.3のブロック図をFigure 3.に示す．今回の再構築により，大きく変わった点を以下にあげる．

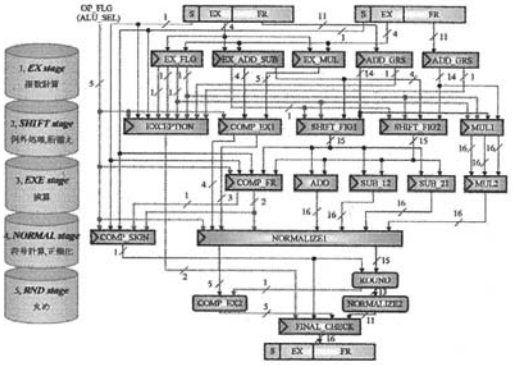


Figure 3. FPU.3 ブロック図

5.1. 乗算

浮動小数点数は符号部，指数部（符号付整数），仮数部（符号なし小数）から構成されている．したがって，浮動小数点数の乗算は，異符号ならマイナス'1'同符号ならプラス'0'である符号計算,指数部を足し合わせる指数計算，仮数部どうしを掛け合わせる仮数計算に分けられる．

これらの処理を行うユニットを新たに組み込むことで，浮動小数点数の乗算を可能とした．

5.2. 例外処理

新たに，指数部が"0000"か"1111"，仮数部が全て'0'であるかどうかを判定するユニットを追加し，それらのユニットから出力されたフラグによって "0×∞"や"∞－∞"等の無効な演算・オーバーフロー・アンダーフローといった例外処理の判定を行うユニットを組み込んだ．

5.3. 漸近的アンダーフロー

漸近的アンダーフローとは，非正規化数を用いることで仮数部の有効桁数を減らす代わりに，アンダーフローの下限を伸ばす仕組みである．この機能を実装するにあたり，仮数の演算結果と指数の演算結果を比較して，指数部が"0000"になる，または仮数部のMSBが'1'になるように指数を減らしつつ仮数を左シフトさせるような正規化ユニットを設計した．Figure 4.に漸近的アンダーフローの仕組みを示す．

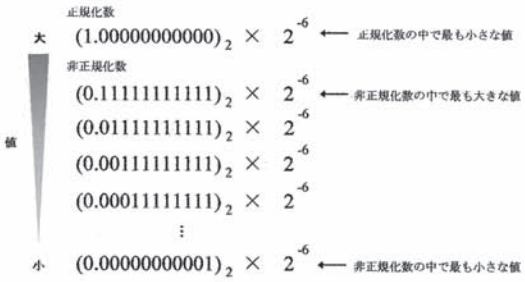


Figure 4. 漸近的アンダーフローの仕組み

5.4. パイプラインの最適化

FPU.2 は EXE stage（演算実行段）が 2 段構成の 7 段パイプライン設計であった．これは，浮動小数点数が“符号＋絶対値”という表現形式のために，加算命令であっても実際の処理は減算であったり，減算命令でも加算処理を行う必要があるためである．FPU.2 では先に仮数を比較して，加減算どちらを行うかを判定してから演算を実行していた．

そこで FPU.3 ではこれらの，仮数の比較・加算・減算を並列に行い，後続のユニットでどの値を利用するか判別するように最適化を行った．また，遅延時間の短い ROUND ユニットや NORMALIZE2 ユニットをひとまとまりとして設計し直すことで，パイプライン段数を 7 段から 5 段へと削減することができた．

各ユニットの遅延調整を行った結果，新たに追加した乗算回路は 12bit×12bit であるので，最も遅延時間のかかってしまう回路である．HCgorilla に組み込むために，より高速動作可能な乗算回路とする必要がある．そこで，まず部分積を計算し，それらを足し合わせて乗算結果を求める．この計算を 2 クロックで処理するように最適化することで，大幅に Critical pass を短縮することが可能となった．

6. FPU.3 解析

6.1. 遅延解析

改良された FPU.3 の各ユニットの遅延時間の解析を行った．その結果を Table 1.に示す．

この表から，最も遅延時間のかかっているユニットは NORMAL stage の中の，演算結果の値の正規化を行う NORMALIZE1 ユニットであり，その遅延時間は 2.37 [ns]となっている．これが新アーキテクチャ FPU.3 のクリティカルパスである．

Stages	Units	Max Delay [ns]
EX stage	ADD_GRS1	0.91
	ADD_GRS2	0.91
	EX_FLG	0.54
	EX_ADD_SUB	1.89
	EX_MUL	1.29
SHIFT stage	EXCEPTION	2.15
	COMP_EX1	1.45
	SHIFT_FIG1	2.30
	SHIFT_FIG2	2.27
	MUL1	2.31
EXE stage	COMP_FR	1.89
	ADD	2.31
	SUB_12	2.29
	SUB_21	2.32
	MUL2	2.32
NORMAL stage	COMP_SIGN	1.58
	NORMALIZE1	**2.37**
RND stage	ROUND	0.80
	NORMALIZE2	0.19
	COMP_EX2	0.78
	FINAL_CHECK	0.48

Table 1. FPU.3 遅延解析結果

6.2. 動作周波数

遅延解析結果から，FPU.3 の Critical pass は 2.32[ns]である．また，HCgorilla は Wave Operation 可能範囲が 2.40[ns]～2.98[ns]であり，クロックサイクル 2.50[ns]，動作周波数 400MHz で動作する．Figure 6.に FPU.3 と HCgorilla の動作可能範囲を示す．Figure 6.から FPU.3 は HCgorilla の動作可能範囲を全てカバーする形で動作可能なので，HCgorilla への組み込みのために FPU.3 のクロックサイクルを HCgorilla と同じ 2.50[ns]とし，動作周波数を 400MHz と確定する．

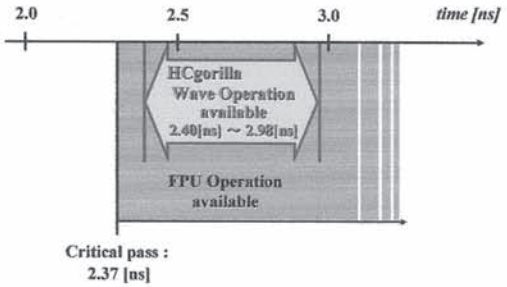


Figure 6. FPU.3 クロックサイクル

6.3. FPU simulation

Figure 7.に FPU.3 の動作シミュレーション結果を示す．2 つの float 型データ・オペレーションコードを入力することで，パイプライン 5 段，動作周波数 400MHz で動作している様子が表れている．また，「乗算」「例外処理」「漸近的アンダーフロー」の処理も正常に動作している．

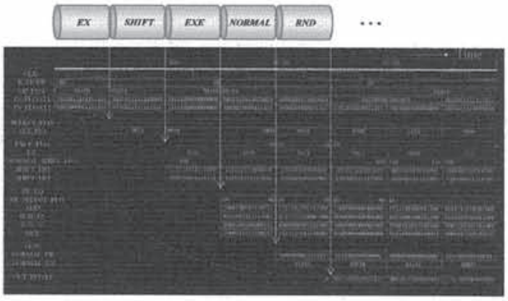


Figure 7. FPU.3 動作シミュレーション結果

7. HCgorilla への FPU 実装

FPU.3 は HCgorilla.3 と同じ 400MHz での高速動作が可能である．従って，遅延時間・動作周波数の差を気にすること無く組み込むことができる．また，JVM の仕様から，float 型演算も他の様々な型の演算同様に，スタック上に積まれたデータに対して演算を行うので，Execute stage に隣接する形で FPU を組み込んだ．新たに FPU を実装したアーキテクチャ HCgorilla.4 のパイプライン構成を Figure 8.に示す．

Figure 8.に示すように int 型演算等を行う Execute stage は2段構成であるのに対し，FPU は5段の構成となっている．このパイプライン段数の違いは，例えば LIW によって int 型命令と float 型命令が並列に処理されるような場合でも，コンパイラによって調整することで，一連の処理の整合性を保ったまま実行することができる．

FPU の実装にあたり，改良された点を以下に上げる．

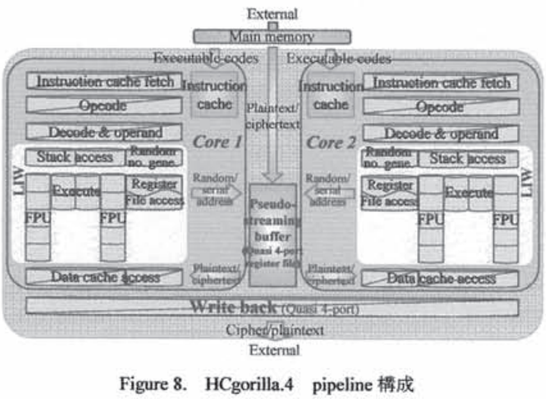


Figure 8. HCgorilla.4 pipeline 構成

7.1. 命令セット拡大

FPU の実装によって，新たに 16 個の float 型 Java instruction が実行可能となる．これにより，従来の 63 命令から 79 命令へと命令セットを拡大した．Table 2.に HCgorilla.4 に搭載された命令セットを示す．

Mnemonic	Binary	Dependent operand (byte)	iconst_0	0x03	0	astore_1	0x4c	0	fconst_0	0x0b	0
			iconst_1	0x04	0	astore_2	0x4d	0	fconst_1	0x0c	0
			iconst_2	0x05	0	astore_3	0x4e	0	fconst_2	0x0d	0
rsw	0xf0	0	iconst_3	0x06	0	pop	0x57	0	fload	0x17	1
rlw	0xf1	0	iconst_4	0x07	0	pop2	0x58	0	fload_0	0x22	0
nop	0x00	0	iconst_5	0x08	0	dup	0x59	0	fload_1	0x23	0
bipush	0x10	1	sipush	0x11	2	dup_x1	0x5a	0	fload_2	0x24	0
iload	0x15	1	aload	0x19	1	dup_x2	0x5b	0	fload_3	0x25	0
istore	0x36	1	iload_0	0x1a	0	dup2	0x5c	0	fstore	0x38	1
iadd	0x60	0	iload_1	0x1b	0	dup2_x1	0x5d	0	fstore_0	0x43	0
isub	0x64	0	iload_2	0x1c	0	dup2_x2	0x5e	0	fstore_1	0x44	0
ineg	0x74	0	iload_3	0x1d	0	imul	0x68	0	fstore_2	0x45	0
ishl	0x78	0	aload_0	0x2a	0	iushr	0x7c	0	fstore_3	0x46	0
ishr	0x7a	0	aload_1	0x2b	0	iflt	0x9b	2	fadd	0x62	0
iand	0x7e	0	aload_2	0x2c	0	ifge	0x9c	2	fsub	0x66	0
ior	0x80	0	aload_3	0x2d	0	ifgt	0x9d	2	fmul	0x6a	0
ixor	0x82	0	astore	0x3a	1	ifle	0x9e	2			
ifeq	0x99	2	istore_0	0x3b	0	if_icmpeq	0x9f	2			
ifne	0x9a	2	istore_1	0x3c	0	if_icmpne	0xa0	2			
if_icmplt	0xa1	2	istore_2	0x3d	0	if_icmpge	0xa2	2	79 命令		
goto	0xa7	2	istore_3	0x3e	0	if_icmpgt	0xa3	2			
iconst_m1	0x02	0	astore_0	0x4b	0	if_icmple	0xa4	2			

Java bytecode-compatible media codes

Table 2. HCgorilla.4 命令セット

7.2. Decode stage の改良

アーキテクチャ HCgorilla.4 では新たに FPU のユニットが加わった事により，Decode stage から出力される各ユニットを制御するためのマイクロプログラムも拡張する必要がある．Table 3.に新たに追加された 16 個の float 型命令のマイクロプログラムを示す．

FPU を制御する為に 4bit 分の制御信号がマイクロプログラムに追加され，従来の 33bit から 37bit の長さへと拡張した．

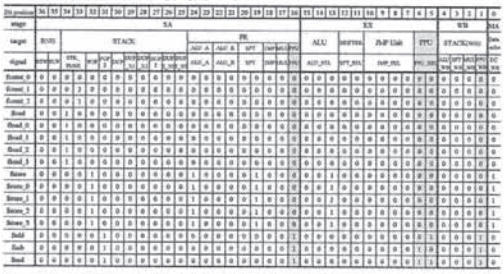


Table 3. HCgorilla.4 追加命令のマイクロプログラム一覧

8. HCgorilla.4 simulation

Figure 9.に HCgorilla.4 の動作シミュレーションに用いたテストプログラムを，Figure 10.にシミュレーション結果を示す．このシミュレーションでは表示の都合上片側コアだけの結果を載せるが，HCgorilla.4 は Multi-core CPU として正常に動作する．

テストプログラムでは，片側コアの一方のパイプで浮動小数点数の乗算，もう一方で浮動小数点数の加算を行う．また，Register file には PPM フォーマットの標準画像が格納されており，

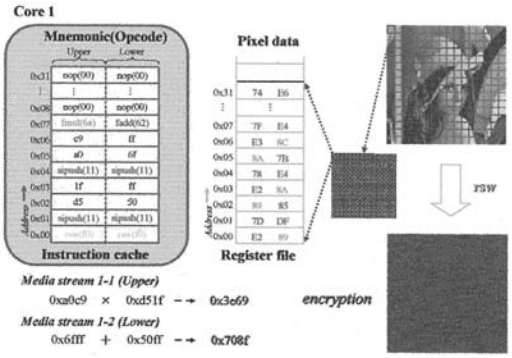


Figure 9. HCgorilla.4 テストプログラム

SIMD mode の暗号化命令 rsw によって，通常の Java instruction を実行しながら，画像データの転置暗号化も同時に行う．

Figure 10.から，スタックに積んだ2 word を pop し，5 clock で演算結果を求め，またスタックへと書き戻してる様子がわかる．浮動小数点数の乗算・加算も正常に計算できている．また，Register file のデータを RNG から出力される擬似乱数と同期させて，画像データを暗号化している様子も現れている．

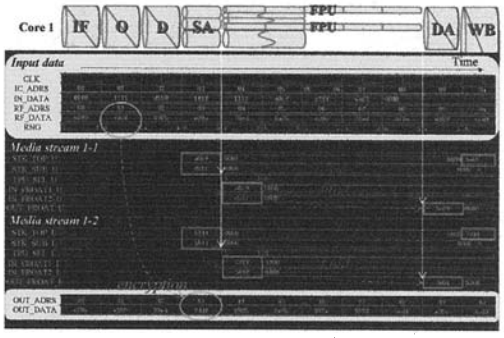


Figure 10. HCgorilla.4 動作シミュレーション結果

9. 結び

我々が開発してきたハードウェア暗号組込み型マルチメディアモバイルプロセッサ HCgorilla に実装するための FPU アーキテクチャの最構築を行い，新たに「乗算」「例外処理」「漸近的アンダーフロー」の機能を追加した．また，パイプラインの最適化を行い，遅延時間の短縮，パイプライン段数の削減を達成した．これにより，動作周波数 400MHz・パイプライン段数 5 段の高速高効率な FPU アーキテクチャが実現された．また，HCgorilla の命令セット拡大・Decode stage の改良により FPU を実装し，新たに 16 個の Java instruction が HCgorilla で実行可能となり，動作シミュレーションにより正常な動作を確認した．

今後の課題としては

- 消費電力・面積の測定
- チップとして試作
- より効率的なパイプライン制御方法の検討

等があげられる．

10. 謝辞

本研究は**弘前大学理工学部長指定重点研究の助成を受けて実施したもので**，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と日本ケイデンス株式会社の協力で行われたものである．

11. 参考文献

[1] 深瀬 政秋，中村 維男，“計算機ハードウェア,”昭晃堂, 1994.

[2] 赤岡 亮，“ハードウェア暗号組み込み型マルチメディアモバイルプロセッサ HCgorilla の設計・開発に関する研究”，弘前大学大学院理学研究科情報科学専攻修士学位論文, 2005.

[3] M. Fukase, R. Akaoka, and T. Sato,“Hardware Cryptography-Embedded Multimedia Mobile Processor for Ubiquitous Computing,” Proc. of 12th NASA Symposium on VLSI Design, pp.1.2.1-1.2.6, Oct. 2005.

[4] Masa-aki Fukase, Hiroki Takeda, Ryo Tenma, Kazunori Noda, Yohei Sato, Ryota Sato, and Tomoaki Sato, “ Development of a Multimedia Stream Cipher Engine,” Proc. of ISPACS 2006, Yonago, Tottori, pp. 562-565, Dec. 2006.2

[5] 三国勝志，“チメディアモバイルプロセッサの開発に関する研究,” 弘前大学理工学研究科修士論文, 2004.

[6] ティム・リンドホルム，フランク・イェリン，“Java仮想マシン仕様 第2版” 村上雅章訳，株式会社ピアソン・エデュケーション, 2001.