平成２１年度戦略的基盤技術高度化支援事業

「Ultra-Android：
マルチコア対応組込みソフトウェア・プラットフォームの研究開発」

研究開発成果等報告書
平成２２年３月

委託者　関東経済産業局
委託先　株式会社つくば研究支援センター

目次

第１章研究開発の概要

１－１　研究開発の背景・研究目的及び目標

【研究背景】

写真＝台湾 HTC 社製 Dream：
米 T-Mobile 最初の Google Android 携帯「G1」を発表（2008 年 09 月 23 日）写真は ITmedia より

次世代携帯電話であるスマートフォンは、ノート PC を超える年間 1 億 8000 万台に達した。そのソフトウェア・プラットフォームは、英 Symian 社が 49%のシェアを持ち優位性は続く。iPhone の 2012 年の市場シェアは 13.6%と楽観的ではないが、Android™が躍進し市場シェアは 18%になると予測されている。

Android™がもたらす携帯端末ソフトウェアのプラットフォーム化およびオープン化により、「ガラパゴス」と呼ばれる日本の携帯電話の作り方が大きく変わる可能性がある。即ち、アプリケーション・ソフトのポータビリティ（移植性）が向上し、多機能化が容易になるとともに、OS 以下の層が非競争領域となり、ハードウェアによる携帯端末機能の差別化や付加価値の創出が困難になる。

現在、携帯端末の性能を左右するプロセッサ(CPU)は、英国 ARM 社の ARM が主流である。半導体ベンダ各社がこの CPU を SoC に搭載する限り大きな性能差はでない。

その結果、半導体も携帯端末も世界的なコスト競争に陥ると予測される。今後、携帯端末ハードウェアで利益を追求するために、ユーザの要求に応える高速化や低消費電力化で大幅に差別化することができる携帯端末用プロセッサ(CPU)と、ソフトウェア・プラットフォームに関するイノベーティブな技術の開発が期待されている。

【研究目的及び目標】

本研究開発は、Android™にヘテロジニアス・マルチコア・プロセッサ技術と分散オブジェクト・ソフトウェア技術を適用し、性能を 1 桁以上向上することで携帯端末や内蔵する半導体等のハードウェアを競争領域に戻し、携帯端末に限らず Android™の採用が見込まれる情報家電の付加価値向上を目的とする。

本研究開発の目標は、Android™用のアプリケーション・ソフトウェアを変更せずに、処理能力が高く消費電力を大幅に抑えられるヘテロジニアス・マルチコア・プロセッサの利用を可能とする「Ultra-Android」ソフトウェア・プラットフォームを開発することである。

「Ultra-Android」ソフトウェア・プラットフォームをトプスシステムズ社のヘテロジニアス・マルチコア・プロセッサ「TOPSTREAM™ Ultra-Android」(開発中)上で動作させた場合に、シングル CPU を用いる携帯端末に対し、以下に示す大幅な性能向上を具体的な目標とする。

① **10 倍以上の高速化**（アプリの処理に掛かる時間が 1/10 以下）
② **10 分の 1 以下の低消費電力化**（アプリの処理に必要な電力が 1/10 以下）
③ **リアルタイム化**（キー入力等に対する応答時間の短縮）

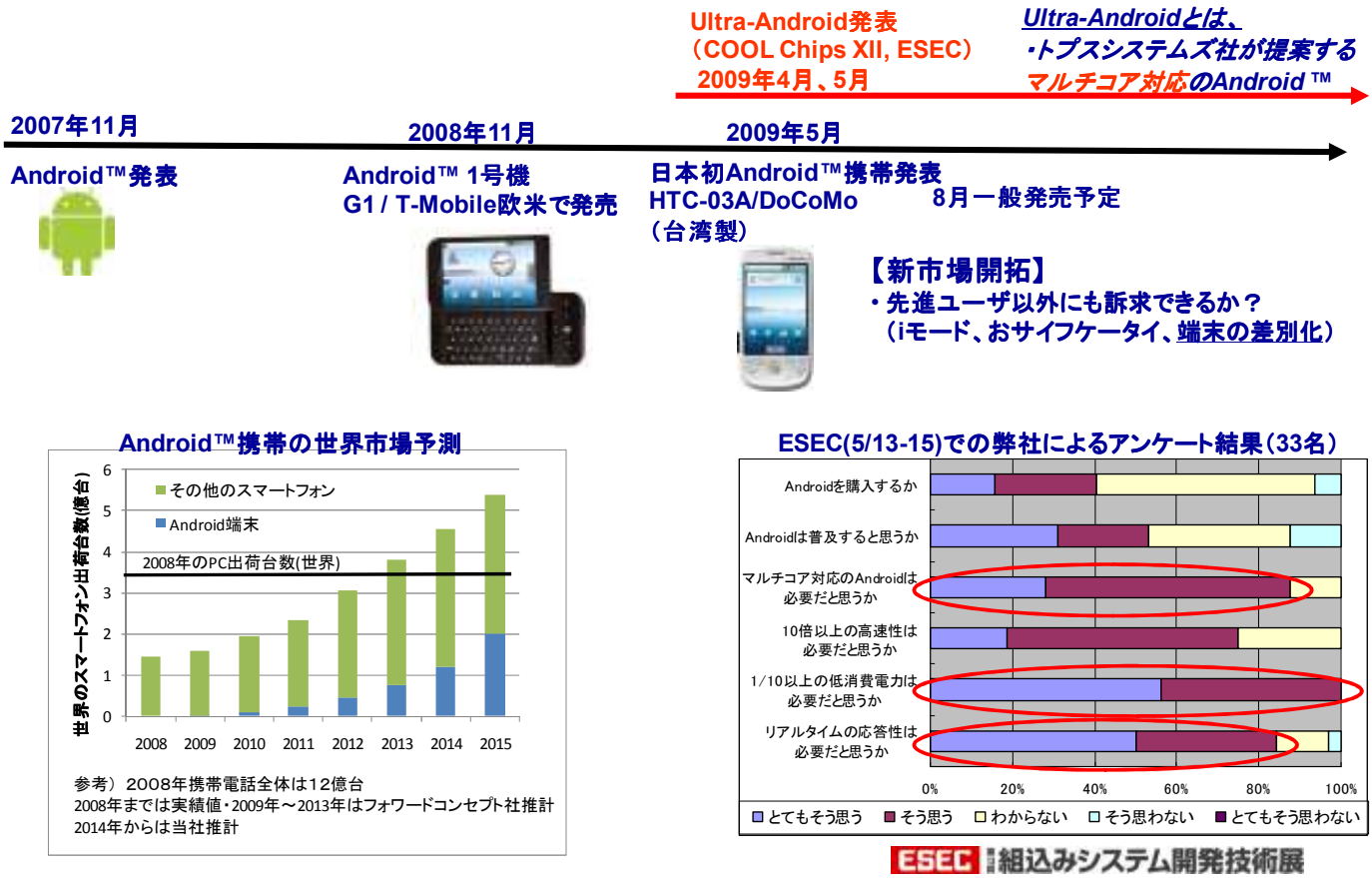
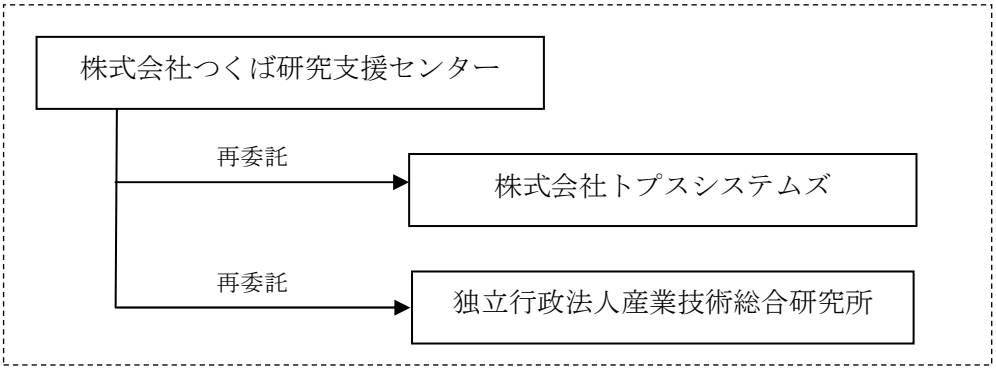


図 1　研究開発の概要　*Android™ への期待と課題*

１－２　研究体制

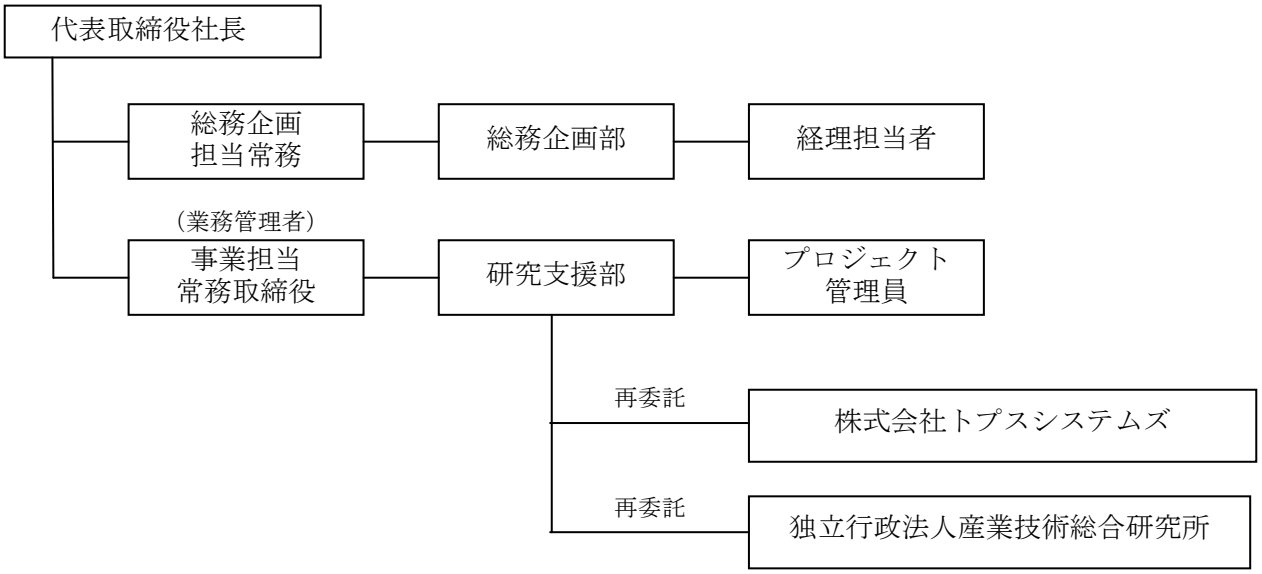
【研究組織（全体）】



【管理体制】

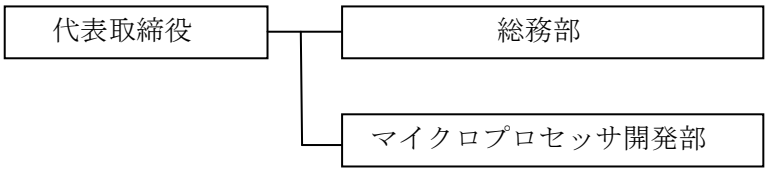
＜事業管理者＞

［株式会社つくば研究支援センター］

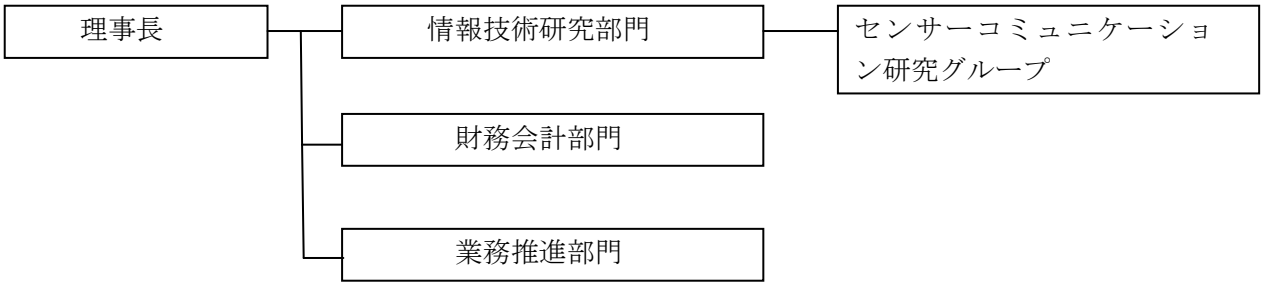


＜再委託先＞

［株式会社トプスシステムズ］



［独立行政法人産業技術総合研究所］



１－３　成果概要

【平成 21 年度の目標】

Android™・Java アプリケーション・プログラムのソースコードを一切変更せずにマルチコア上でソフトウェアを並列に実行できる基本的な例を示し、ソフトウェア並列化とマルチコア上での処理最適化により 10 倍以上に高速化もしくは低消費電力化する「Ultra-Android」が実現可能であることを、マルチコア・プロセッサ・モデルを開発し、システム評価シミュレータを用いて実証すること。

比較対象を Google Android 携帯「G1」（ARM11（528MHz）搭載）とする。また、Android™における処理がマルチコア・プロセッサ上で必ずデッドラインの時間までに終了することを保証する 1ms 単位のハード・リアルタイム性を実現可能な方式を開発すること。

尚、各目標値の比較対象＜基準機＞は、《最初の Google Android 携帯「G1」》とする。

＜基準機＞

台湾 HTC 社製　Dream　（Google Android 携帯「G1」）

参考 URL：　http://htcdream.com/

〔スペック（抜粋）〕

・メインチップ Qualcomm 社製 MSM7201A

・アプリケーションコア：ARM11@528MHz

・メモリ Samsung 社製 K5E2G1GACM

・256MB　NAND フラッシュメモリ

・128MB　DDR SDRAM

評価項目	A（TOPS担当） ソフトウェアの 並列化	B（TOPS担当） ソフトウェアの 処理速度向上	C（TOPS担当） ソフトウェアによる 低消費電力化	D（産総研担当） ソフトウェアによる リアルタイム化
評価指標	ソースコード変更	処理速度向上	消費電力削減	リアルタイム性
基準機 ： 台湾HTC社製Dream （Google Android™携帯「G1」）	不可 （単一CPU）	1.0	1.0	無し
【平成21年度】 1. Android™の分析 1-1. 分析環境の構築 1-2. Android™アプリケーションの分析 1-3. Android™プラットフォームの分析 *性能10倍を達成する ための見通しを得る*	分析環境の構築完了 実行ステップ数を1/10とする機能分散手順の明確化 リアルタイム性を確保する手順の明確化 （Android™アプリケーション、Android™ソフトウェアプラットフォーム）			
	並列化 分析＆検討	高速化 分析＆検討	低消費電力化 分析＆検討	リアルタイム化 分析＆検討
【平成22年度】 2. Ultra-Androidソフトウェア・ プラットフォームの開発 2-1. 仕様設計 2-2. 評価環境の構築 2-3. プラットフォーム開発（実装） 2-4. ソフトウェア性能評価 ＜エミュレーション環境での評価＞	部分的に人手に よるソースコード 変更の必要あり	2倍以上	2分の1以下	数ミリ秒程度
【平成23年度】 3. 評価システムの開発 3-1. 評価システム仕様設計 3-2. 評価ボード作成 3-3. システム性能評価 ＜FPGA評価ボード上での評価＞	0% （人手によらない 自動化を実現）	10倍以上	10分の1以下	1ミリ秒以下

【平成 21 年度の目標達成状況】

平成 21 年度は研究開発の初年度ということで、先ずは高速化対象及び低消費電力化対象の Android™アプリケーションを選定して分析し、オブジェクト分散によるソフトウェアの並列化方式を開発した。また、「Android™」用の既存のアプリケーション・ソフトウェアを「Ultra-Android」を用いて並列化して、「TOPSTREAM™ Ultra-Android」上で動作させた場合の処理速度向上、消費電力削減、リアルタイム性向上の効果について、システムレベルのソフトウェアおよびハードウェアの性能モデルを用いた性能シミュレーションにより実証した。

Ⓐソースコードを変更しない「Android™」ソフトウェア並列化方式の確立

「Android™」プラットフォーム上で動作する Java アプリケーション・ソフトウェアのソースコードを一切変更せずに、マルチコア上でアプリケーション・ソフトウェアを並列に実行するためのオブジェクト分散処理によるソフトウェア並列化方式を確立した。

Ⓑ「Android™」ソフトウェアの処理速度向上の課題への対応

「Android™」ソフトウェアのオブジェクト分散処理による並列化と「TOPSTREAM™ Ultra-Android」上でのソフトウェア間の通信とソフトウェア実行処理の最適化により、シングル CPU での処理に比べ 10 倍以上の高速化が可能である事をソフトウェアの性能モデル及び「TOPSTREAM™ Ultra-Android」の性能モデルを作成し、システム評価シミュレータ（VisualSim）を用いて実証した。

Ⓒ「Android™」ソフトウェアによる低消費電力化の課題への対応

「Android™」ソフトウェアをシングル CPU で処理する場合に比べ、ソフトウェア並列化と「TOPSTREAM™ Ultra-Android」上でのソフトウェア実行処理の最適化により 10 分の 1 以下に低消費電力化が可能である事を、ソフトウェアの性能モデル及び「TOPSTREAM™ Ultra-Android」の性能モデルを作成し、システム評価シミュレータ（VisualSim）を用いて実証した。

Ⓓ「Android™」ソフトウェアにおけるリアルタイム化の課題への対応

「Android™」ソフトウェアを「TOPSTREAM™ Ultra-Android」上で実行する際に、ある処理が必ず決めたデッドライン時間までに終了するようなハード・リアルタイム処理が可能なオブジェクト分散処理によるソフトウェア並列化方式を開発した。

具体的な「Android™」ソフトウェアのリアルタイム性の評価対象として、タッチパネルディスプレイへの入力から対応するソフトウェアの呼び出しまでの応答時間を評価し、応答速度 1ms を「TOPSTREAM™ Ultra-Android」で実現するためのハード・リアルタイム処理方式を開発した。

Ⓔ「Android™」ソフトウェア・プラットフォーム基本性能の確認

「Android™」ソフトウェアを動作させるために必要となるコンパイラ・Linux カーネル・基本ライブラリをヘテロジニアス・マルチコア上で動作可能にするための仕様を作成して移植した。

第２章　本論

「Android™」のアプリケーション・ソフトウェア及びソフトウェア・プラットフォームの動作を「Android™」携帯端末及びソフトウェア分析環境を構築して分析し、「Ultra-Android」ソフトウェア・プラットフォームとして並列化・高速化・低消費電力化・リアルタイム化を実現するオブジェクト分散処理による並列化方式を開発した。

そして、既存の「Android™」アプリケーション・ソフトウェアを、「TOPSTREAM™ Ultra-Android」上で動作する「Ultra-Android」ソフトウェア・プラットフォームを用いて分散処理させた場合の処理速度向上・消費電力低減・リアルタイム性向上等の効果をシステム・レベルの性能シミュレーションにより実証した。

図２に本研究開発が対象とした Android™ ソフトウェア・プラットフォームの構成要素を示す。

図２　Ultra-Android　組込みソフトウェアプラットフォームの研究開発

本年度の研究開発は以下及び図３に示す手順で行った。

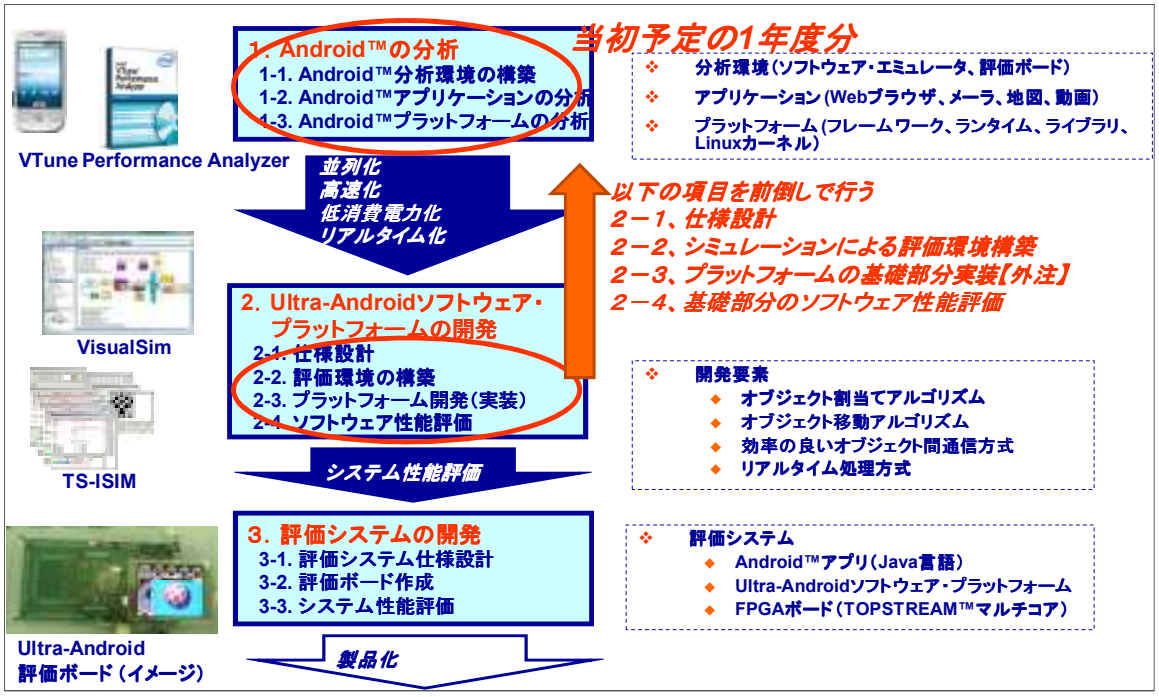


図３　研究開発の方法

２－１　①.「Android™」の分析

分析の目的毎に適切なアプリケーションを選定した（表 1）。

表 1　Ultra-Android 化の対象とするアプリケーションの選定

	分析の目的	アプリケーションの使われ方	備考
キー入力 （autotext）	・高速化 ・低消費電力化 ・リアルタイム性	・一定間隔（100ms～1s）のキー入力に対するGUI画面への文字出力 ・メールアプリを想定	・単純な例として基本的な分析に用いる
ゲーム （モグラたたき）	・低消費電力化 ・リアルタイム性	・タッチパネル入力に対する画面応答 ・一定間隔での画面更新（10fps-60fps）	・秒間フレーム数の要件が満たせれば、それ以上高速化する必要はない
WEBブラウザ	・高速化	・画面スクロール ・数秒～数分に1回のWEBページ更新	

キー入力は、携帯電話のメール文字入力アプリケーションを想定した。現状の携帯電話では、キー入力の応答速度を向上させるために、プロセッサを高速で動かしているため、電力消費が大きい。そのため、高速化、低消費電力化、リアルタイム性を追求する対象として選定した。

ゲームは、定期的な画面更新に関わる消費電力が大きく、また高速なユーザ入力への応答が求められる。そのため、低消費電力化とリアルタイム性を追求する対象として選定した。

WEB ブラウザは、画面のスクロール表示や WEB ページの読み込み時に高速性が要求される。そのため、高速化を追求する対象として選定した。

２－１－１　①－１．「Android™」分析環境の構築

Android™アプリケーション（Java）を実行してアプリケーション及びソフトウェア・プラットフォームの実行状況をプロファイリング分析するための分析環境を構築した。

Ⅰ．実ハードウェア実行による分析環境

基準機である Android Dev Phone 1、及び Android Dev Phone2、ARM 搭載の EVM（アットマーク社製 Armadillo500FX 及び OMAP35 EVM 評価ボード）を用いて、分析環境を構築した（図 4、図 5）。

図 4　Android Dev Phone 1 と TraceView を用いた分析環境

図 5　Armadillo500FX と RealView を用いた分析環境

Ⅱ．ソフトウェアエミュレーションによる分析環境

QEmu(Android™標準の PC 上のエミュレータ)を用いて、分析ソフトウェアを構築した（図 6）。

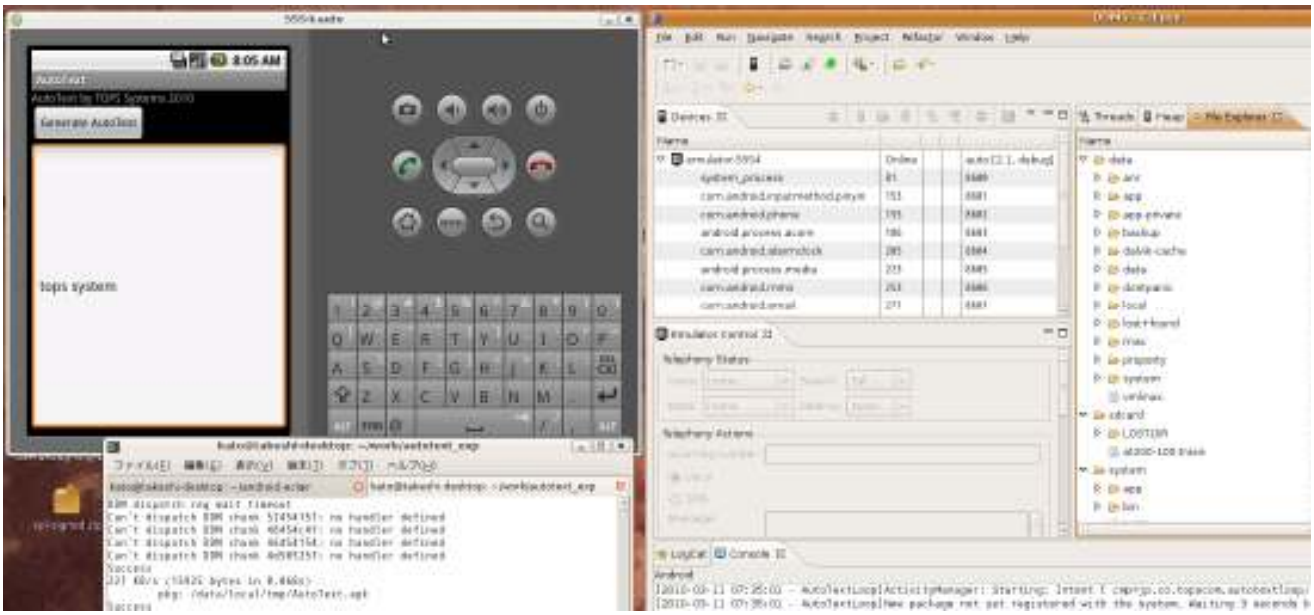


図 6　ソフトウェアエミュレータ(QEmu)を用いた分析環境　（画面写真）

Ⅰ．及びⅡ．の分析環境で使用した Android™分析ツールを表 2 に示す。

表 2　Android™分析ツール一覧

	分析項目	分析対象	分析粒度	手法	備考・制約条件
Oprofile	関数滞在時間	Linuxシステム全体	C言語関数	サンプリング（パフォーマンスモニタ使用）	サンプリング手法のため、メソッド呼び出し回数は取得不可能
	呼び出し回数	Linuxシステム全体	C言語関数		
	メモリアクセス量	Linuxシステム全体	C言語関数		
TraceView	メソッド滞在時間	Android™/Javaアプリケーション	Javaメソッド呼び出し	トレース	Javaアプリ内にDebug.startMethodTrace
	コールグラフ	Android™/Javaアプリケーション	Javaメソッド呼び出し		
ddms	スレッド状態（utime, stime）	Android™システム全体	スレッド	dalvikVMデバッグポート接続	
	VM ヒープ使用量	Android™システム全体	DalvikVM		ヒープ使用量取得時にGCを起こす必要がある
	オブジェクト種類別VM ヒープ使用量	Android™システム全体	Javaオブジェクト		
	オブジェクトアロケーション追跡	Android™システム全体	Javaオブジェクト		
ps	メモリ占有量	Linuxシステム全体	Linuxプロセス	Linuxプロセステーブル参照	/proc以下のファイルを読み出しても等価

Ⅲ．リアルタイム性分析

ARM マイコン搭載の Android™開発ボードのバスをモニターし、FPGA の回路との協調動作も可能な「ソフト分析用デバッグ装置・リアルタイム性評価用特注 FPGA ボード」を開発し、時間測定を行った（図 7、図 8、図 9)。

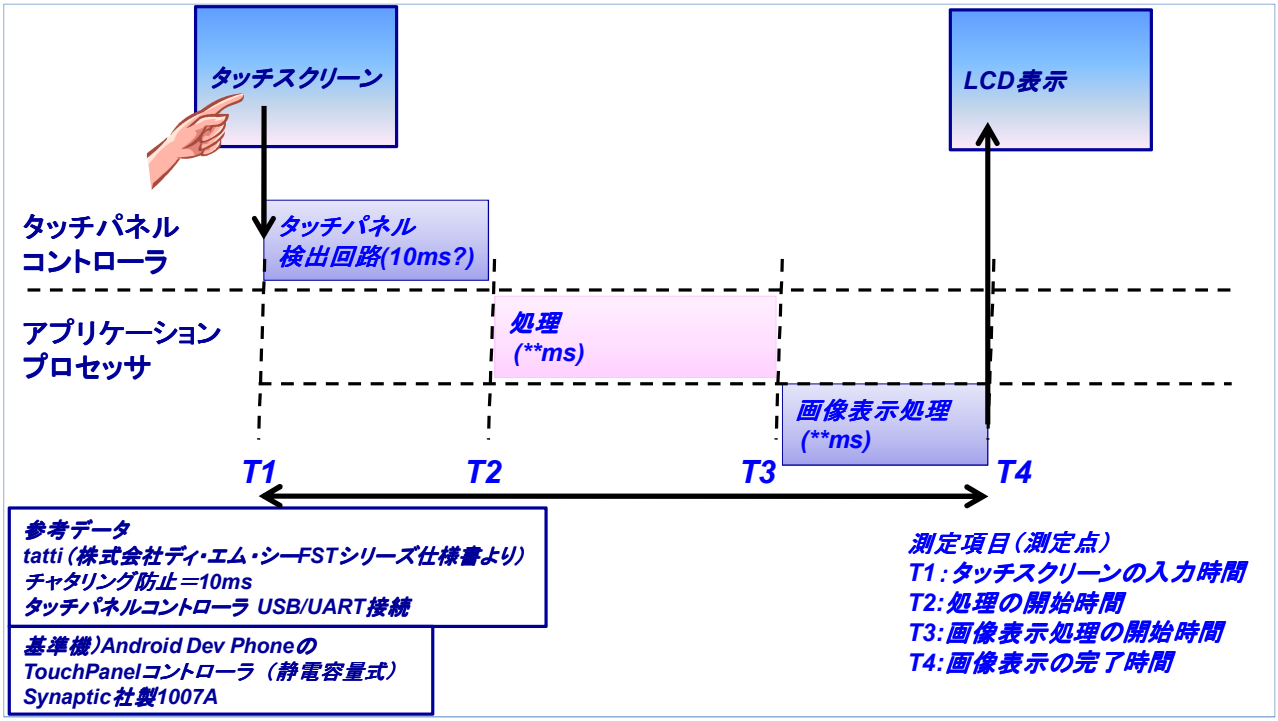


図 7　タッチパネル入力からの応答時間評価概要

図８　リアルタイム性分析環境　（写真）

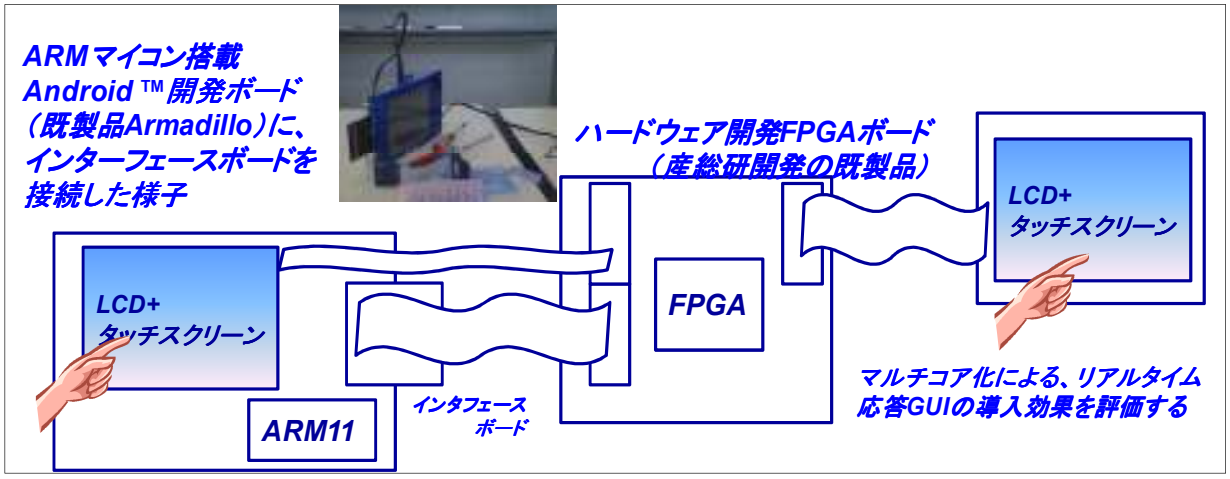


図９　リアルタイム性分析環境の構成

２－１－２　①－２．「Android™」アプリケーションの分析
及び ①－３．「Android™」プラットフォームの分析

Android™アプリケーション及びAndroid™プラットフォームの分析は、以下の手順で進めた。

◆プリ分析
◆フェーズ１分析（機能分散処理の適用性を見極めるための分析）
◆フェーズ２分析（高速化、低消費電力化、リアルタイム化のための分析）

表３　Android™アプリケーション分析マトリックス

		分析の目的	分析手法	分析対象										
				アプリケーション										
				テキスト入力			ゲーム			Webブラウザ				
	プリ分析で、以降のPhase1の分析項目の絞り込みを行う			キー入力	処理	表示	キー入力	処理	表示	入力	処理	通信	処理	表示
準備	テストケース設定		n/a	文字入力サンプル作成			ゲーム自動実行			テスト用Webサイト作成				
分析 Phase-I	プリ分析	分析対象の絞り込み	処理時間プロファイル	○	○	○	○	○	○	○	○	○	○	○
	機能分散処理の適用性分析	・オブジェクト処理のボトルネックを洗い出す ・呼出し関係の局所性を洗い出す	オブジェクト生成分析	○	○	○	○	○	○	○	○	○	○	○
			処理呼出し分析（コールグラフ）	○	○	○	○	○	○	○	○	○	○	○
			処理時間分析（スレッド単位）	○	○	○	○	○	○	○	○	○	○	○
分析 Phase-II ①	Phase１で、Phase２以降の分析項目の絞り込みを行う 高速化分析	・演算処理上のボトルネックを洗い出す ・メモリ・アクセス上のボトルネックを洗い出す ・オブジェクト間通信上のボトルネックを洗い出す	処理内容分析（演算タイプ）	○	○	○				○	○	○	○	○
			各種並列化可能性分析	○	○	○				○	○	○	○	○
			細粒度プロセス分析	○	○	○				○	○	○	○	○
			メモリアクセス分析（R/W）	○	○	○				○	○	○	○	○
			メモリ局所性分析	○	○	○				○	○	○	○	○
			KPN化可能性分析	○	○	○				○	○	○	○	○
			通信・同期分析	○	○	○				○	○	○	○	○
			複合命令化可能性分析	○	○	○				○	○	○	○	○
			ストリーム処理可能性分析	○	○	○				○	○	○	○	○
分析 Phase-II ②	低消費電力化分析	・プロセッサの動作周波数をどこまで低減できるかを見極める	処理内容分析（演算タイプ）	○	○	○	○	○	○					
			メモリアクセス分析（R/W）	○	○	○	○	○	○					
			メモリ局所性分析	○	○	○	○	○	○					
			分散処理適用性分析	○	○	○	○	○	○					
			複合命令化可能性分析	○	○	○	○	○	○					
			ストリーム処理可能性分析	○	○	○	○	○	○					
分析 Phase-II ③	リアルタイム化分析	・応答性能を上げるためのボトルネックを洗い出す ・ボトルネックの並列化可能な部分を洗い出す	リアルタイム要求分析	○	○	○	○	○	○					
			ボトルネック分析	○	○	○	○	○	○					
			分散処理適用性分析	○	○	○	○	○	○					
			並列処理適用性分析	○	○	○	○	○	○					

【分析結果】

＜テキスト入力アプリ＞

モジュール単位の処理時間と割合を図 10 および表 4 に示す。モジュール単位で見ると、dalvik VM（libdvm.so）が全体の 38.9%、カーネル（vmlinux）が全体の 31.5%と突出していることが分かる。

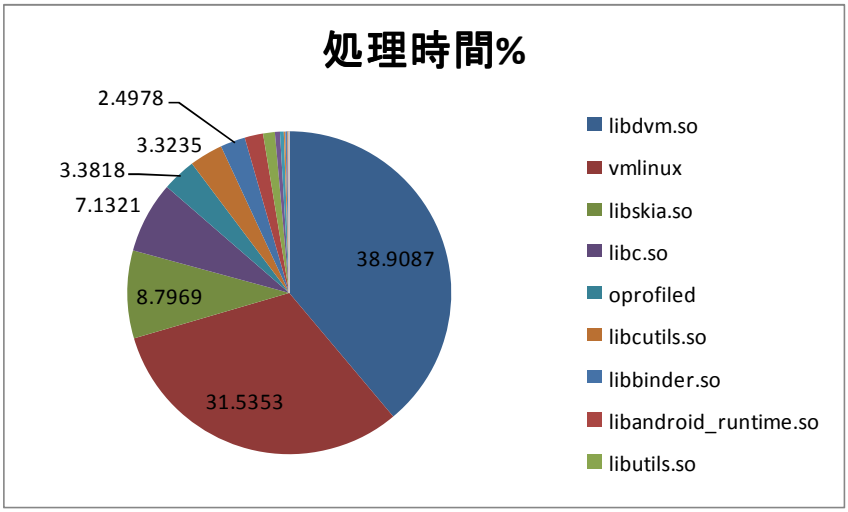


図 10　プリ分析結果：モジュール別処理時間　（テキスト入力アプリ）

表 4　プリ分析結果：モジュール別処理時間（テキスト入力アプリ）

サンプル数	%	累積%	モジュール
3592048	38.9087	38.90870	libdvm.so
2911337	31.5353	70.44400	vmlinux
812130	8.79690	79.24090	libskia.so
658439	7.13210	86.37300	libc.so
312205	3.38180	89.75480	oprofiled
306830	3.32350	93.07830	libcutils.so
230599	2.49780	95.57610	libbinder.so
168326	1.82330	97.39940	libandroid_runtime.so
108834	1.17890	98.57830	libutils.so
47482	0.51430	99.09260	libui.so
34031	0.36860	99.46120	libsurfaceflinger.so
15273	0.16540	99.62660	libGLES_android.so
15029	0.16280	99.78940	libm.so
5073	0.05500	99.84440	libEGL.so
4444	0.04810	99.89250	copybit.msm7k.so
2299	0.02490	99.91740	gralloc.msm7k.so
2026	0.02190	99.93930	libpixelflinger.so
1653	0.01790	99.95720	liblog.so
1543	0.01670	99.97390	libGLESv1_CM.so
1076	0.01170	99.98560	libicuuc.so
558	0.00600	99.99160	libstdc++.so
443	0.00480	99.99640	libicui18n.so
242	0.00260	99.99900	libnativehelper.so
38	0.00041	99.99941	adbd
19	0.00021	99.99962	linker

C 関数毎の処理時間分析結果を表に示す。

0.01%以上の処理時間を占める関数は、2900 個である。処理時間の多い順の上位には、libskia の Dither 関数（画面表示に関わる）、libcutils の android_memset32 関数等がある。

表 5　プリ分析結果：C 言語関数毎の処理時間分析結果（テキスト入力アプリ）

サンプル数	%	累積%	ファイル情報	モジュール	C 言語関数名
1730210	19.0950	19.0950	InterpC-portdbg.c:1458	libdvm.so	dvmInterpretDbg
734457	8.1056	27.2006	board-sapphire.c:479	vmlinux	h2w_config_cpld
511182	5.6415	32.8421	InterpC-portdbg.c:1380	libdvm.so	checkDebugAndProf
504527	5.5681	38.4102	SkBlitRow_D16.cpp:146	libskia.so	S32A_D565_Opaque_Dither(unsigned short*, unsigned int const*, int, unsigned int, int, int)
217539	2.4008	40.8110	InterpAsm-armv5te.S:409	libdvm.so	dalvik_inst
211077	2.3295	43.1405	(no location information)	vmlinux	modedb
187692	2.0714	45.2119	memset32.S:55	libcutils.so	android_memset32
133028	1.4681	46.6800	Profile.c:628	libdvm.so	dvmMethodTraceAdd
130351	1.4386	48.1186	InterpAsm-armv5te.S:9791	libdvm.so	common_invokeMethodNoRange
108945	1.2023	49.3209	wait.c:44	vmlinux	remove_wait_queue
101932	1.1249	50.4458	db_insert.c:60	oprofiled	odb_update_node_with_offset
80627	0.8898	51.3356	memory.c:1891	vmlinux	do_wp_page
70845	0.7819	52.1175	twofish_common.c:584	vmlinux	twofish_setkey
70650	0.7797	52.8972	vfprintf.c:300	libc.so	vfprintf
64604	0.7130	53.6102	Thread.c:2421	libdvm.so	dumpWedgedThread
56929	0.6283	54.2385	board-trout-panel.c:381	vmlinux	trout_mddi_power_client
51040	0.5633	54.8018	MarkSweep.c:526	libdvm.so	scanObject
48546	0.5358	55.3376	(no location information)	libdvm.so	_dvmHeapSourceStartup_veneer
46894	0.5175	55.8551	dlmalloc.c:4221	libc.so	malloc
43309	0.4780	56.3331	InterpAsm-armv5te.S:9980	libdvm.so	common_returnFromMethod
42036	0.4639	56.7970	TypeCheck.c:243	libdvm.so	dvmInstanceofNonTrivial
..	…	…	…	…	…

C 言語関数毎の処理時間を、上位 140 関数について累積した結果を図 11 に示す。処理時間比率が 0.5%以上の関数は 31 個で、全処理時間の 56%をカバーすることが分かる。上位 100 関数では全体の 73%をカバーする。

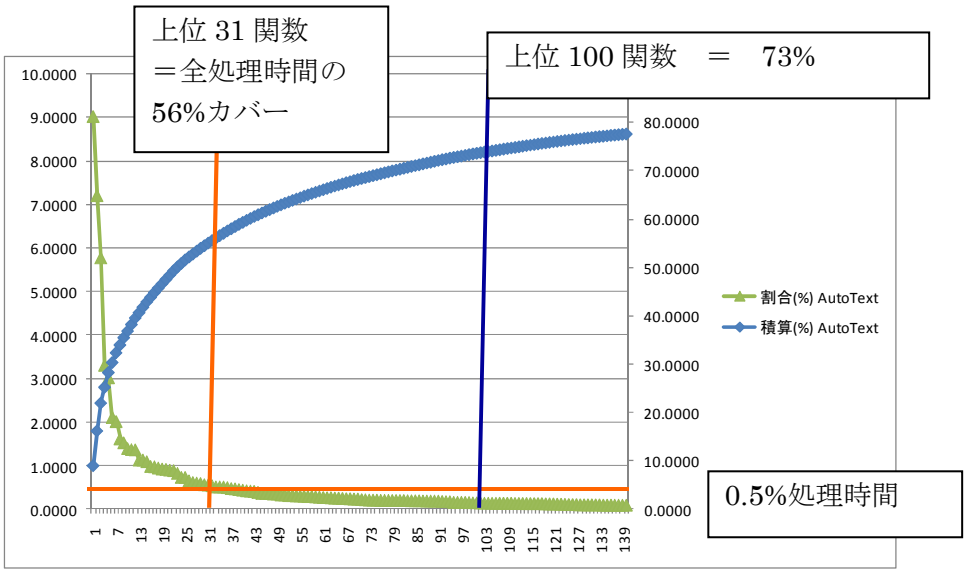


図 11　関数別処理時間を上位 140 関数について累積した結果（テキスト入力アプリ）

TraceView で取得したコールグラフを図に示す。点線で示す２つの部分にわけた左側の部分は「キー入力処理」に関わるイベントの処理であり、右側の部分は「描画」に関わるイベントの処理である。

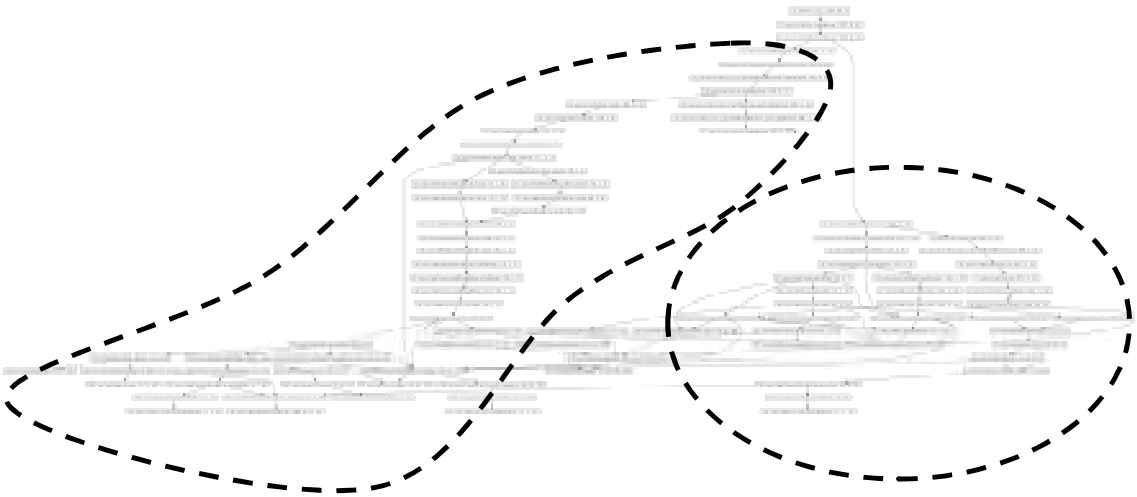


図 12　Java メソッド単位のコールグラフ　（テキスト入力アプリ）

２－２　②．Ultra-Android ソフトウェア・プラットフォームの開発

２－２－１　②－１．仕様設計

Android™をマルチコア対応とするための Ultra-Android プラットフォーム仕様を設計した。

【高速化・低消費電力化方式】

分析結果に基づいて、Android™ソフトウェア・プラットフォームの各構成要素に対する高速化・低消費電力化方式を図に示す。

図 13　オブジェクト　高速化方式

◆Ultra-Android ソフトウェア・プラットフォームの構成（図 14）

Ultra-Android では、機能分散処理のために表 6 のコンポーネントを追加する。

表 6　分散処理ミドルウェア（Distributed Processing Middleware）

分類	コンポーネント名	機能説明
Distributed Processing Middleware	Micro ORB Engine	TOPSTREAM™ Ultra-Android のプロセッサコア間の通信機能を最大限に活用し、Dalvik での Java オブジェクトのリモート呼び出しを効率的に行うためのモジュール
	Static Mapper	Java オブジェクトの型情報と実行するプロセッサコアの静的マッピングを事前の指定に基づいて、Java オブジェクトを TOPSTREAM™ Ultra-Android 上の指定するプロセッサに配置して実行指示をするモジュール
	Dynamic Mapper	Java オブジェクトの実行時情報とプロセッサコアの負荷状況等に基づいて、Java オブジェクトを TOPSTREAM™ Ultra-Android 上のプロセッサに配置して実行指示をするモジュール
Android Runtime	Remote Call Extension	Dalvik における Java オブジェクトのリモート呼び出し機能を拡張し、Micro ORB Engine を使用する為のコンポーネント

図 14　Android™と Ultra-Android ソフトウェア・プラットフォームの構成

◆Ultra-Android で変更するコンポーネントとその内容

表 7 に、Ultra-Android 仕様 Rev.1 の時点での変更予定コンポーネントを示す。

表 7　Android™と Ultra-Android のソフトウェア構成比較表

分類	コンポーネント	Ultra-Android での変更点	Android での役割説明
Application Framework	Activity Manager	アクティビティ状態に応じたプロセッサ配置のスケジューリング	アプリケーションのライフサイクルを管理
	Window Manager	ウィンドウごとの機能分散化	ウィンドウを管理
	Contents Manager	→	アプリケーション間のデータ共有を管理
	View System	機能分散化	ユーザインターフェイスを管理
	Notification Manager	リアルタイム情報表示のための機能分散化	ステータスバーへのアラート表示を管理
	Package Manager	→	インストールを管理
	Telephony Manager	→	電話機能の管理
	Resource Manager	→	非コード資源を管理
	Location Manager	→	位置情報の管理
	XMPP Service	→	メッセージングサービスを管理
Android Runtime	Core Libraries	TOPSTREAM™のアーキテクチャのストリーム処理機能を活用したパフォーマンスライブラリを使用（算術演算・メモリ転送等）	Java 言語に準拠したコアライブラリ機能
	Dalvik VM	豊富なレジスタによる高速実行 メソッド呼び出し高速化	.dex 形式のバイトコードを実行する仮想マシン

Libraries	Surface Manager	→	複数アプリケーション間の 2D/3D 画像合成
	Media Framework	TOPSTREAM™のアーキテクチャのストリーム処理機能を活用したパフォーマンスライブラリを使用	ビデオ形式の再生と記録
	SQLite	→	リレーショナルデータベース
	OpenGL ES	→	3D グラフィックス・エンジン
	Free Type	→	ビットマップとベクタフォントのレンダラ
	WebKit	TOPSTREAM™のアーキテクチャのストリーム処理機能を活用したパフォーマンスライブラリを使用	ブラウザ表示を行うための HTML レンダラ
	SGL	→	2D グラフィックスエンジン
	SSL	→	セキュアな通信ライブラリ
	libc	→	標準的な C 言語ライブラリ
Linux Kernel	Display Driver	デバイス毎担当コアで機能分散	ディスプレイデバイス
	Camera Driver	デバイス毎担当コアで機能分散	カメラデバイス
	Bluetooth Driver	デバイス毎担当コアで機能分散	Bluetooth デバイス
	Flash Memory Driver	デバイス毎担当コアで機能分散	フラッシュメモリデバイス
	Binder (IPC) Driver	マイクロ ORB エンジンで実現	共有メモリによるプロセス間通信（2ms 程度）
	USB Driver	デバイス毎担当コアで機能分散	USB デバイス
	Keypad Driver	デバイス毎担当コアで機能分散	キーパッドデバイス
	WiFi Driver	デバイス毎担当コアで機能分散	Wifi 無線 LAN デバイス
	Audio Driver	デバイス毎担当コアで機能分散	音声デバイス
	Power Manager	デバイス毎担当コアで機能分散	電源管理デバイス

２－２－２　②－２．評価環境の構築

Ultra-Android ソフトウェア・プラットフォームの性能を評価するためのシミュレーション環境を構築する。仮想HWプラットフォームとして、システムレベル性能評価シミュレータ (VisualSim) を用い「TOPSTREAM™ Ultra-Android」プロセッサの性能モデルを用いた評価環境を構築した（図 15）。

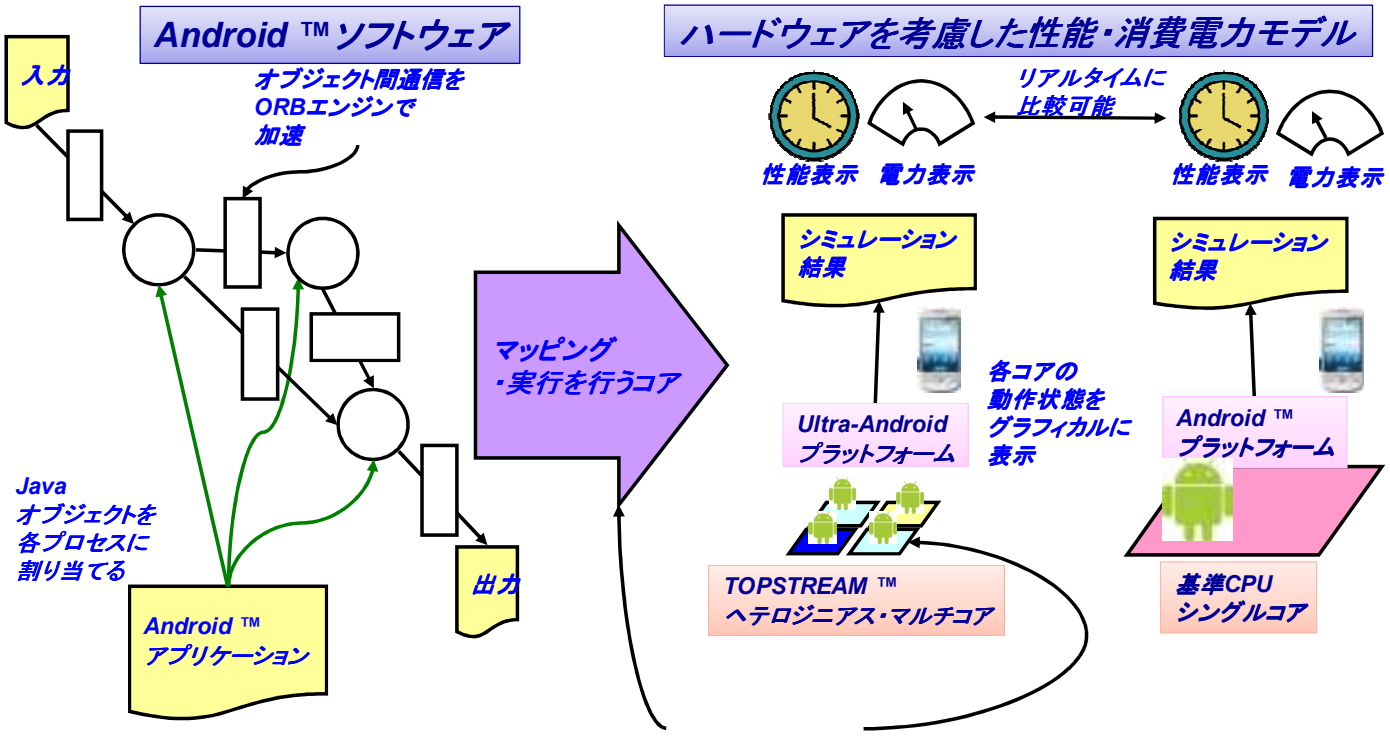


図 15　Ultra-Android 性能・消費電力シミュレーションの方法

【評価環境の詳細】

VisualSim 上に構築した TOPSTREAM™ Ultra-Android 性能・消費電力評価モデルを図 16 に示す。

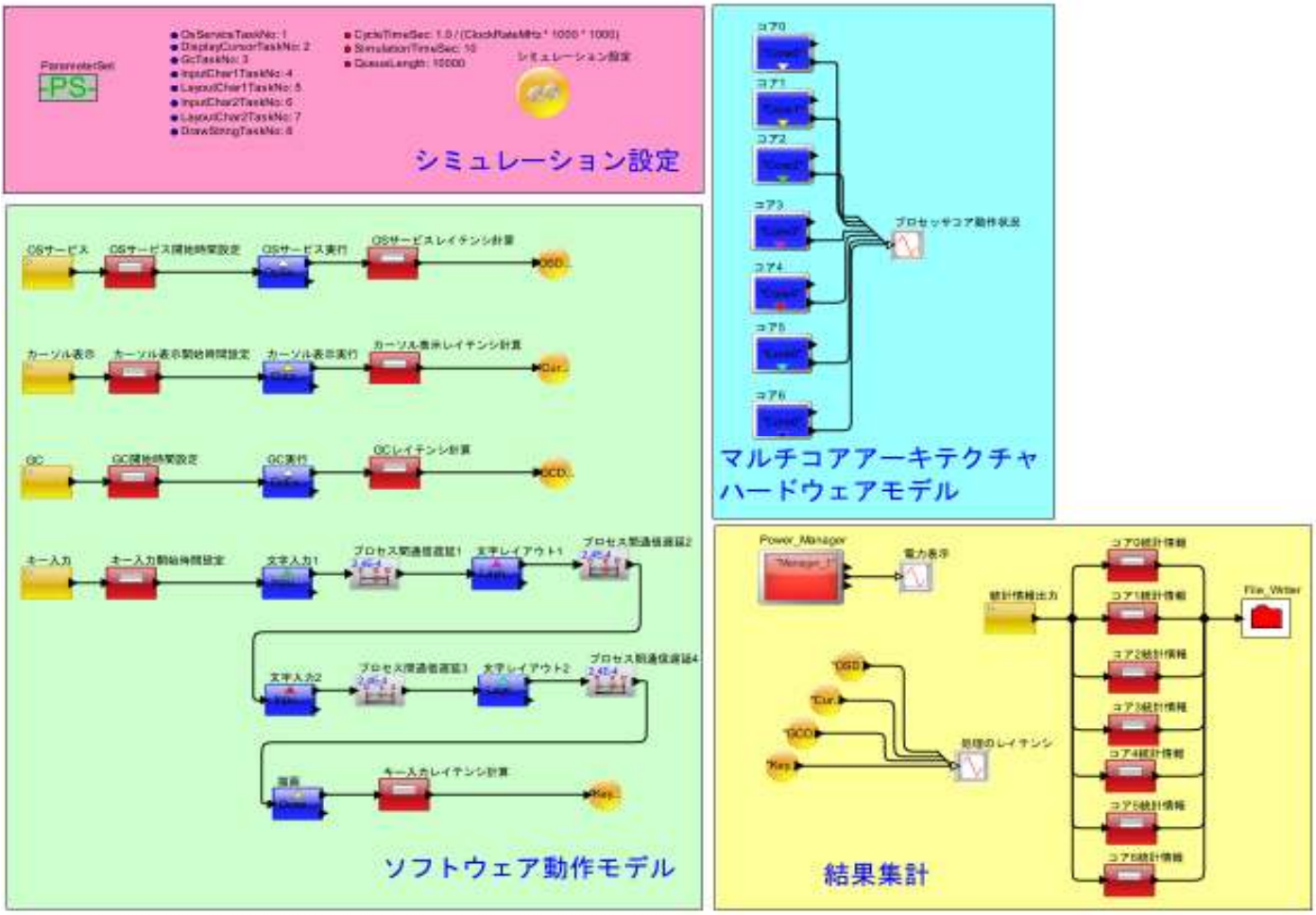


図 16　TOPSTREAM™ Ultra-Android 性能・消費電力評価モデル

【評価結果】

・消費電力：　基準 CPU に対して約 1/25

・処理速度：　基準 CPU に対して 10 倍（この時の消費電力は 1/5）

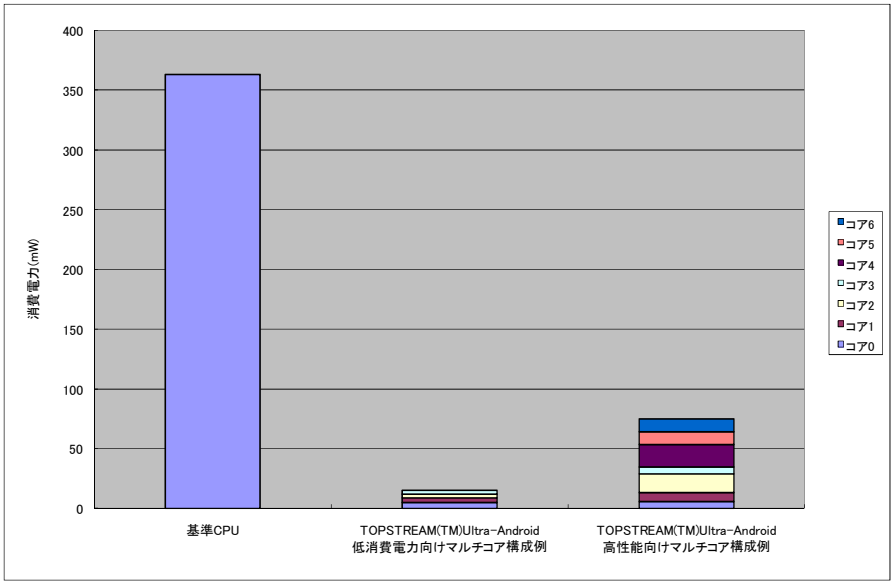


図 17　各構成に対する消費電力シミュレーション結果

表 8　各構成に対する消費電力の相対比

	基準CPU	TOPSTREAM™Ultra-Android	
		TOPSTREAM™Ultra-Android低消費電力向けマルチコア構成例	TOPSTREAM™Ultra-Android高性能向けマルチコア構成例
消費電力(mW)	362.9	14.7	74.3
相対比	100.0%	4.1%	20.5%
	2467.3%	100.0%	505.0%
	488.6%	19.8%	100.0%

２－２－３　②－３．プラットフォーム開発（実装）

Ultra-Android の仕様に基づき、プラットフォーム開発の実装を進めた

【「Linux 用コンパイラ」性能評価結果】

Ultra-Android 実現のため、Linux 用コンパイラ gcc および GNU ツールチェイン binutils の TOPSTREAM™ MC プロセッサ（TOPSTREAM™ Ultra-Android 用マルチコアの制御用 CPU）に移植した。

性能評価は、Linux 用コンパイラ gcc により Linux カーネル内ソースコードをビルドし、生成されたオブジェクトコードのステップ数を指標とした。

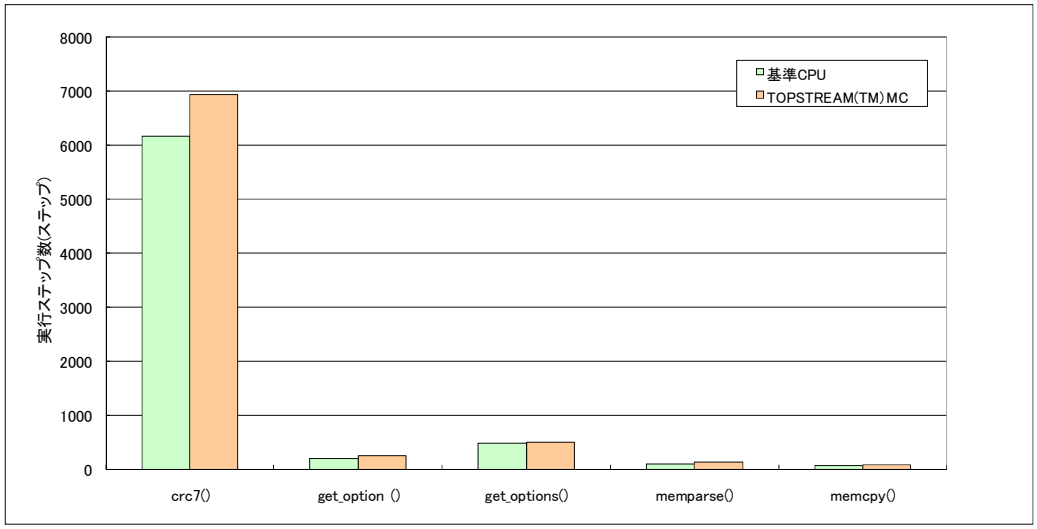


図 18　Linux カーネル内の関数の実行ステップ数の比較

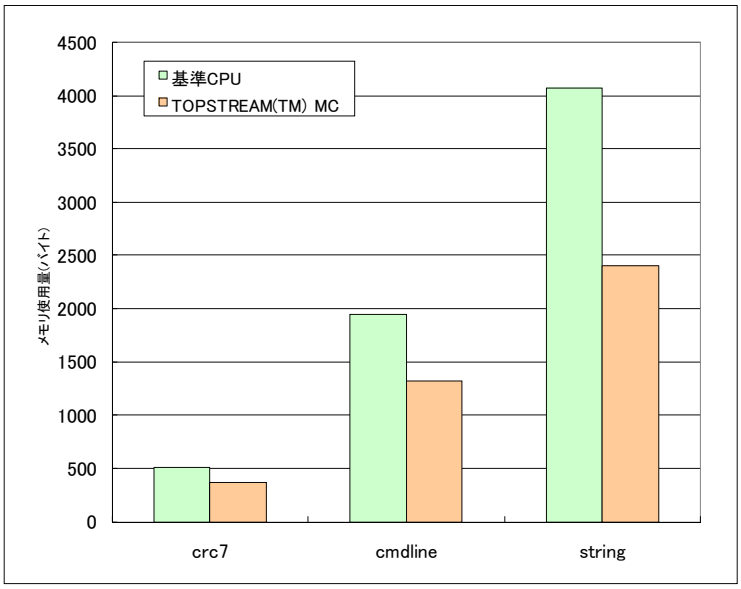


図 19　Linux カーネル内関数のメモリ使用量の比較

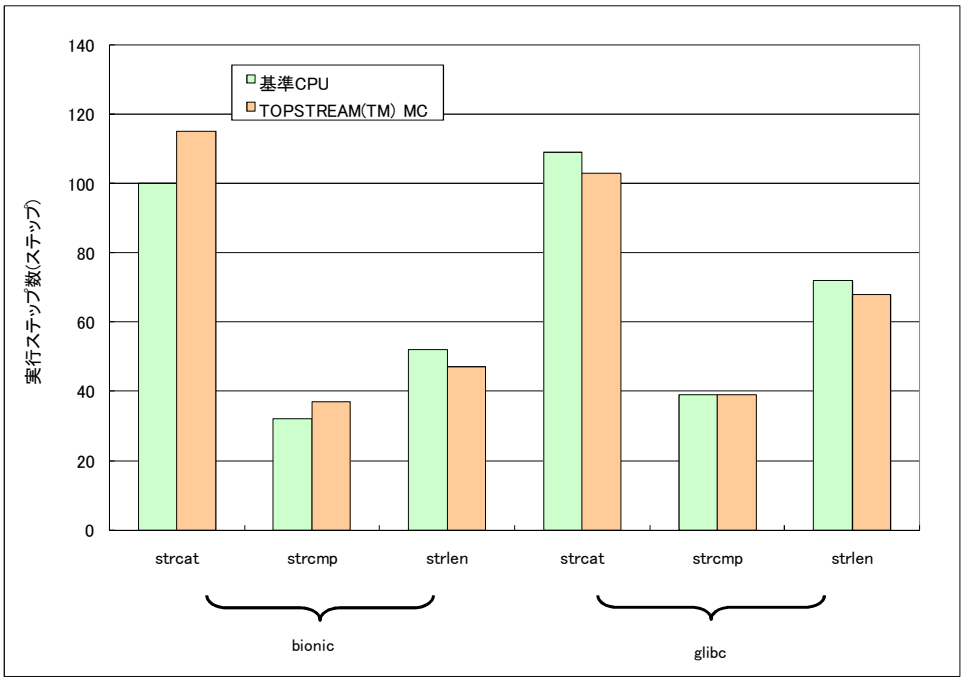


図 20　bionic および glibc 内の関数の実行ステップ数の比較

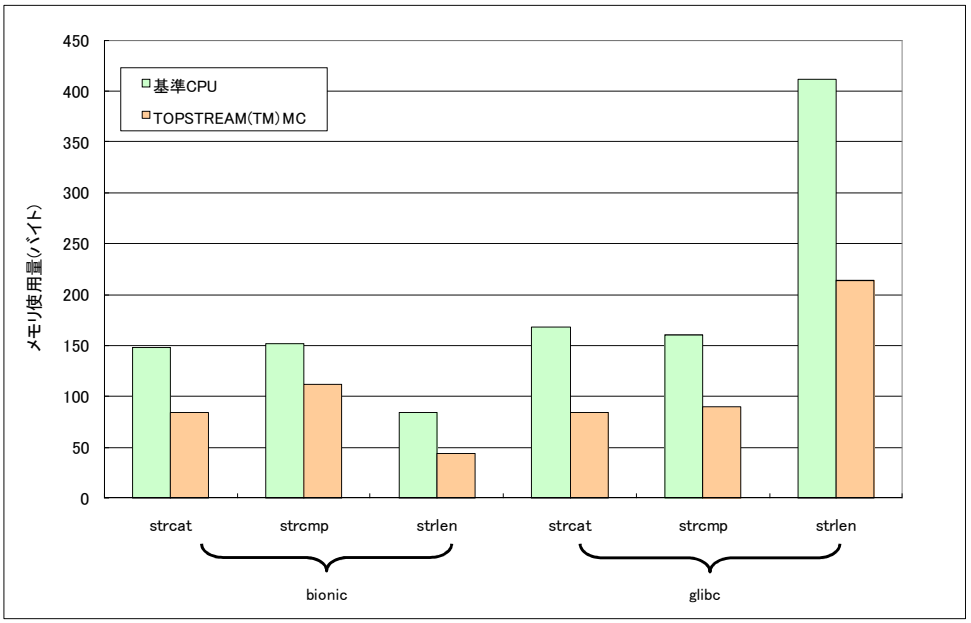


図 21　bionic および glibc 内の関数のメモリ使用量の比較

２－３　　専門用語等の解説

【Android™】携帯電話用ソフトウェアのオープンソース・プラットフォーム。ミドルウェア、ユーザーインターフェース、電話帳などの標準的なアプリケーション・ソフトウェアを含んでおり、Windows Mobile や Symbian OS に近い。Linix がカーネルに採用されているが、カーネル以外は Linux 関連技術を使用していない。

【マルチコア】１つのプロセッサ LSI 内に複数のプロセッサ・コアを集積する技術。プロセッサの動作周波数の向上が困難になってきたため、プロセッサの数を増やすことで性能向上を目指す方式。同一のプロセッサ・コアを複数個集積するホモジニアス・マルチコアと、異なる種類のプロセッサ・コアを複数個集積するヘテロジニアス・マルチコアとがある。

【ヘテロジニアス・マルチコア】異種のアーキテクチャをもつマイクロプロセッサを統合したもの。プロセッサ・コアをアプリケーションの処理に特化することにより、ある動作周波数で達成できる演算処理性能を高めることができる。ヘテロジニアス・マルチコア上のソフトウェア処理方式は、機能分散処理になる。

【機能分散処理】機能毎に専用機に分散して処理を行う方式。

【オブジェクト指向】オブジェクト間の相互作用としてシステムの振る舞いをとらえる考え方。

第３章　全体総括

本研究開発により、次の成果を得ることができた。

1.「Android™」のアプリケーション・ソフトウェア及びソフトウェア・プラットフォームの機能分散処理による並列化・高速化・低消費電力化・リアルタイム化に有用な分析結果を得た。

2.「Ultra-Android」ソフトウェア・プラットフォーム仕様(Rev.1.0)を策定した。

本研究開発の完了は平成 23 年度であるが、平成 22 年度中には本研究開発の成果を活用した携帯端末のカスタム設計案件を受託することを目標とする。

【Ultra-Android プラットフォームを用いた製品の市場と売り上げ目標】

Android 端末は、既に 16 機種が発売されており、更に 8 機種以上の発売が予定。（LG, Sony Ericson, HTC, ASUS, Samusung, Garmin, Huawei, Motorola, 東芝、シャープ等。）Android™のスマートフォン市場における成長速度は、他の対抗する OS を大幅に上回り、2013 年には世界で第 2 位のシェアを占めるまでになると予測（IDCC が 2010 年 1 月 25 日に発表した調査レポート）されている。世界の携帯電話市場は 2007 年に 11.5 億台である。そのうち、スマートフォンの市場は景気後退期においても成長しており、世界の出荷数は 2009 年に 13%成長し、1 億 6400 万台に達すると予測されている（米国の調査会社フォワードコンセプト社）（図 22）。また、Gartner の 2009 年 10 月の予測によると、2013 年の Android 搭載機は、7,600 万台に達する見通しである。

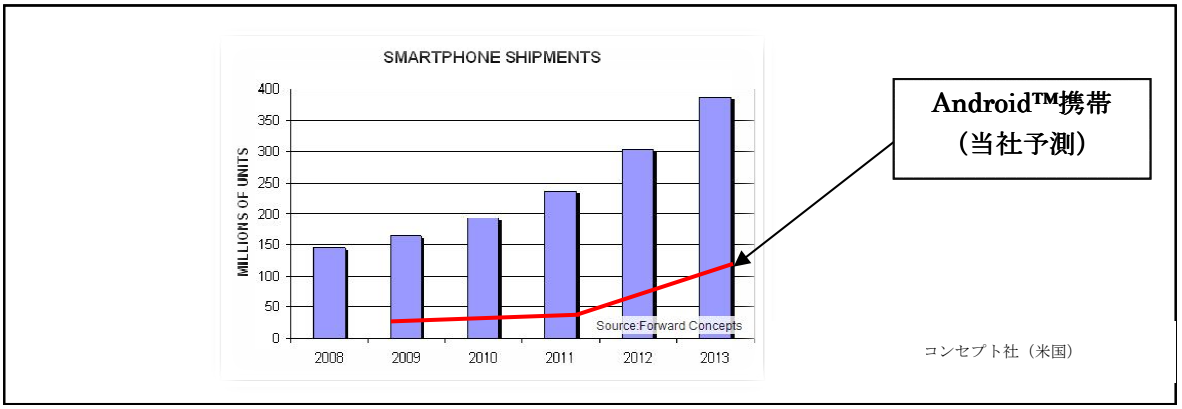


図 22　スマートフォン市場予測

本研究開発成果である「Ultra-Android」は、上述の Android™搭載機の普及の中で予測される処理性能不足やバッテリの持ちに対するユーザの不満を解消する手段として、先ずは国内のスマートフォン・メーカと協力して国内市場をターゲットに製品化し、その後世界市場を目指す。

＜ビジネス・モデル＞

・ハードウェア IP 製品及びソフトウェア IP 製品のライセンシング販売

－TOPSTREAM™Ultra-Android プロセッサ（ハードウェア IP）

－Ultra-Android ソフトウェア・プラットフォーム（ソフトウェア IP）【本研究開発の成果】

注：IP 普及のための戦略として、TOPSTREAM™ ソフトウェア・プラットフォーム IP のオープン・ソース（無償化）も検討している。オープン化時には、IP の価格を調整するとともに、スマートフォン・メーカ毎に TOPSTREAM™ ソフトウェア・プラットフォームのカスタム化サービスを提供する予定である。