（様式 5）

指導教員 承認印	

平成　23 年　7 月　29 日

学位（博士）論文の和文要旨

論文提出者	工学府博士後期課程電子情報工学専攻 平成 20 年度入学 学籍番号　08834302　氏名　　太 田　　淳　　印
主指導教員 氏　　名	中條　拓伯
論 文 題 目	Dalvik アクセラレータ: Android 端末における Java アプリケーションの高速実行機構

論文要旨（2000 字程度）

「Android」のアプリケーション実行基盤である「Dalvik VM」には，実行速度に課題がある．また Dalvik VM における高速化手法は乏しい現状にある．本論文では，Android アプリケーションの高速実行を実現する機構を提案する．1 つ目は Dalvik VM が実行するバイナリ「Dalvik バイトコード」を入力，プロセッサの命令セットに変換する「Dalvik バイトコード・アクセラレータ」である．2 つ目は，Dalvik バイトコードにおいて演算対象である Dalvik レジスタのロード・ストア削減機構「DRMT (Dalvik Register Map Table)」である．3 つ目は Dalvik バイトコード・アクセラレータと Dalvik VM 間のモード遷移コストを削減する機構である．この 3 つの機構により，VM での実行や JIT に比べ効率的な命令の生成・実行を示す.

第 1 章「緒言」では， Android の著しい市場拡大を示した．一方で，アプリケーションの実行環境である Dalvik VM の実行速度には課題があることを述べた.現在の Dalvik VM は Java VM に比べて高速化手法に乏しく，動作速度を向上させる意義があることを述べた．それを踏まえ，ハードウェアにより Android アプリケーションのバイナリである Dalvik バイトコードを直接実行する，Dalvik バイトコード・アクセラレータを提案，その有効性を示すことを本論文の目的として設定した.

第 2 章「Java VM」では，Java VM の動作について述べた．Java においてはすでに高速化手法が複数存在している．本論文で実装する手法を紹介する手前，Java VM の内部アーキテクチャ，動作原理について紹介した.

第 3 章「Java における高速化手法」では，既存の Java VM における高速化手法を述べた．ソフトウェアによる手法としては，JIT，AOT(Ahead Of Time)コンパイルを挙げた．いずれ

の方法も，生成したネイティブ・コードを保持するために使用する主記憶・補助記憶が増加することが課題である．ハードウェアによる手法は，プロセッサとの接続関係によって，コプロセッサ型，データ共有型，制御共有型の 3 種類に大分できる．各実装方式のうち制御共有型の「Jazelle DBX」が回路規模の目安となるゲート数が少なく，組み込み機器向けに適していることを挙げ．Dalvik バイトコード・アクセラレータの実装方式として用いることを述べた．

第 4 章「Dalvik アーキテクチャ」では，Dalvik VM の内部アーキテクチャ，動作，Dalvik バイトコードについて述べた．Dalvik VM はレジスタベースの VM であり，VM が実行するバイナリも Java とは異なる．章末にて Java VM との相違点を示し，Dalvik VM の高速化手法がまだ不十分であることを述べた．

第 5 章「Dalvik バイトコード・アクセラレータ」では，本論文で提案するアクセラレータの仕様と動作を述べた．まず，Jazelle DBX を基として，MIPS アーキテクチャのプロセッサ・パイプラインのフェッチ・ステージとデコード・ステージの間に，Dalvik デコーダを搭載することを示した．Dalvik デコーダは，Dalvik バイトコードをデコードしプロセッサのネイティブ・コードに変換する．続いて Dalvik デコーダの内部構造と，どのようにして Dalvik バイトコードを入力，変換し，ネイティブ・コードを生成する過程を述べた．

第 6 章「DRMT」では，Dalvik VM やバイトコード・アクセラレータの生成する命令には，メモリのロード・ストアの無駄が生じていることと，問題の解決手法として DRMT を用いる手法について述べた．Dalvik VM の演算対象である Dalvik レジスタは，メモリ上に存在する配列である．特に高速化を行っていない VM やクセラレータでは，バイトコード単位でメモリのロード・ストアが発生することとなる．DRMT により複数の Dalvik バイトコードを跨いで，ロードした Dalvik レジスタを物理レジスタに保持する．そして DRMT により Dalvik レジスタのロード・ストアを削減する動作例を示した．

第 7 章「遷移コストの削減機構」では，Dalvik バイトコード・アクセラレータと Dalvik VM の間で実行モードの遷移が発生する際，大きなサイクルを要していることを示した．これは Dalvik バイトコード・アクセラレータの性能向上を損ねる．これに対し，アクセラレーション可能，不可能なバイトコードの連続を予測し，モード遷移を減らす，高速に各モードが用いるレジスタを切り替えるレジスタウィンドウを提案，評価した．

第 8 章「プロセッサシミュレータ上での Android 実行」では，Dalvik バイトコード・アクセラレータを実装し，性能を評価する環境として用いるプロセッサシミュレータ「SimMips」について，その概要とアクセラレータを搭載するのに必要な変更を述べた．変更を加えた SimMips 上でベンチマーク・プログラムを実行し，実際の携帯端末向けプロセッサに近い特性あることを示した．

第 9 章「評価」では，本論文で示したアクセラレータならびに DRMT について，高効率な命令生成を評価しその結果を述べた．評価は 2 種類行った．1 つ目は DRMT により Dalvik バイトコードから，不必要な Dalvik レジスタのロード・ストアが削減されているか評価した．2 つ目はアクセラレータが生成した命令について JIT と比較し，効率的な命令生成が行われているかを評価した．

第 10 章「結論」では，本論文の結論を述べた．まずアクセラレータならびに DRMT により，VM や JIT に比べより効率的な命令生成が行われる要因を述べた．次に今後の展望として，SimMips やハードウェアへのアクセラレータの実装の可能性と，それにより得られる評価について述べた．