様式 C-19

科学研究費補助金研究成果報告書

平成２３年　６月１３日現在

機関番号：６２６１５
研究種目：基盤研究（C）
研究期間：2008～2010
課題番号：２０５０００５９
研究課題名（和文）　SVG 記述のハードウェア描画装置に関する基礎的研究

研究課題名（英文）　A Fundamental Study on Hardware Accelerator for SVG

研究代表者
米田　友洋（YONEDA TOMOHIRO）
国立情報学研究所・アーキテクチャ科学研究系・教授
研究者番号：３０１８２８５１

研究成果の概要（和文）：本研究では，SVG 記述をストリームとして入力し，それを直接解析し，並列パイプラインを用いて低位描画命令列への変換を行うハードウェアの構成方法について検討し，具体的にハードウェアを設計し，論理シミュレーションによりその機能を確認した．また，ハードウェアにより効率良くデコードを行うために，SVG 記述中の描画要素のうち，並列に描画しても描画結果には影響を与えないものを抽出する前処理ソフトウェアツールを作成した．

研究成果の概要（英文）：In this research project, we have investigated a hardware accelerator for SVG, which accepts SVG descriptions as streams, analyzes them directly, and translates them into low-level drawing command sequences based on a concurrent pipelining mechanism. This idea has been implemented as a logic circuit, and it has been functionally tested using a logic simulation tool. Furthermore, in order to decode SVG descriptions efficiently by hardware, a software preprocessor which extracts sets of objects that can be concurrently drawn without affecting the final results has been developed.

交付決定額

（金額単位：円）

	直接経費	間接経費	合　計
２００８年度	1,100,000	330,000	1,430,000
２００９年度	500,000	150,000	650,000
２０１０年度	1,900,000	570,000	2,470,000
年度			
年度			
総　計	3,500,000	1,050,000	4,550,000

研究分野：情報工学
科研費の分科・細目：情報学・計算機システム・ネットワーク
キーワード：非同期式回路設計，SVG，ハードウェア化

１．研究開始当初の背景

(1) SVG(Scalable Vector Graphics) は，World Wide Web Consortium の勧告として公開された，XML ベースの 2 次元ベクトル画像記述言語であり，特徴として，スケーラブルできれいな画像を表現できる，テキスト形式なので管理が容易である，などがあげられる．最近では，そのサブセットが第 3 世代携帯電話の仕様の一部として採用されており，SVG 記述の描画を高速に，かつ，CPU に負荷をかけずに低消費電力で実行する方式への要望が高まっている．

(2) SVG 記述を直接描画するには，SVG 記述の解析(デコード) と低位描画命令列の生成という 2 つのステップが必要となる．前者は，ハードウェアパターンマッチング技術を応用することにより可能である．ストリームに

対するマルチパターンのマッチングアルゴリズムとしては AC 法[1] がよく知られており，それに基づいたハードウェア実現方法がいろいろ提案されている(例えば[2])．SVG 記述のデコードには，タグの開始と終了を検出し，親子関係，兄弟関係を抽出する必要があるが，AC 法における AC オートマトンを適切に構成し，上記と同様なハードウェア化を行うことで実現可能であると思われる．ただし，検出し終わったタグ・属性(直線や円などの描画の種類を表す) や属性値(線の長さや色などを表す) から低位描画命令列を効率よく生成する技術，および，兄弟関係等により得られる並列描画の可能性を生かして高速描画する技術が課題となる．我々は，ハードウェアパターンマッチングアクセラレータの基礎的研究を行っており，その中で得られた，状態遷移機械のハードウェア実現技術および非同期式パイプラインを用いたデータフロー処理技術を改良・応用することで，これらの課題を解決できると考えた．

[1] A.V. Aho and M.J. Corasick, Efficient String Maching: An Aid to Bibliographic Search, Communications of the ACM, 18(6); pp.333-340, 1975.

[2] N. Tuck, T. Sherwood, B. Calder, and G. Varghese, Deterministic Memory-Effi-cient String Matching Algorithms for Intrusion Detection, Proc. IEEE Infocom, pp.333-340, 2004.

２．研究の目的

(1) 本研究では，SVG 記述をストリームとして入力し，それを直接解析し，並列性を抽出した上で，並列パイプラインを用いて低位描画命令列への変換を行うハードウェアの構成方法について検討する．

(2) 具体的には，SVG 記述の効率的デコード方法，低位描画命令列の効率的生成方法，パイプライン機構の効率的実現方法，必要並列度に応じたハードウェア構成，ボトルネックと性能評価等を検討し，SVG 記述のハードウェア描画装置の実現可能性およびその有効性を明らかとする．なお，本提案は基礎的研究でもあり，SVG 記述の範囲としては，SVG Tiny(http://www.w3.org/TR/2003/REC-SVGMobile-200301 14/) の小さなサブセットを扱う．

３．研究の方法

(1) SVG デコード用状態遷移機械のハードウェア化としては，遷移を引き起こす入力を 8 ビットとした場合，256 個の遷移先状態エントリを並べておき，現状態エントリアドレスに入力ビットを連接したもので，遷移先状態エントリを引くことにより，高速に状態遷移が実現できる(図１)．

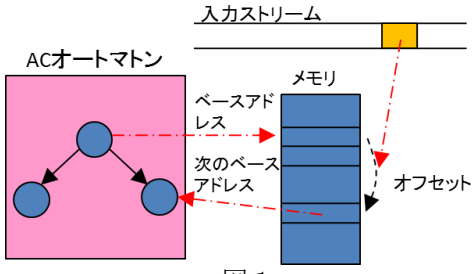

図１

しかし，この方法では遷移先数が少ない場合，多くの無駄エントリが生じる．遷移入力が 8 ビットでも，平均的には遷移先は 256 個よりも遙かに小さいことを考慮し，各状態において実際に遷移を引き起こす入力の各ビットの共通の部分を mask として記録し，異なる部分のみのエントリをメモリ上に並べる．例えば，

入力１	1110 0011	offset	入力１	1101 1
入力２	1001 0000		入力２	0010 0
入力３	1100 0010		入力３	1001 0
mask	1*** 00**			

ただし，mask されないビット(offset)のビット数が大きくなると無駄なエントリが増えるため，offset の例えば上位 4 ビットについて，そのパタンが実際に表れるかどうかを 16 ビットのビットマップ(bitmap)として持つ．上記の例では，３つの入力に対し，以下のように bitmap を作る．

	0123	4567	89ab	cdef
bitmap	0010	0000	0100	0100

この例では，bitmap を使わないと，offset が 5 ビットであるため，メモリ上に 32 エントリが作られ，そのうち 3 エントリが使われる(29 エントリが無駄) のみであるが，bitmap を使うことにより，6 エントリ(offset の残りが 1 ビットであるため) が作られ，3 エントリのみが無駄になる．offset は最大 8 ビットとなり得，その場合最悪 1 遷移ごとに 15 エントリ(offset の残りが 4 ビットとなるため)が無駄になる．逆に，offset が 4 ビット以下の場合，bitmap により完全にエントリが記述されるため，無駄エントリは 0 となる．メモリ上の各エントリには，遷移先エントリとともに mask と bitmap を格納しておくことにより，遷移先でも同様な処理が行える．典型的な SVG 記述を用いて，平均的にメモリ量が少なくなるように offset 長を決める．

(2) SVG のデコードが終了すると，描画コマンドに対応する「タグ・属性」と描画データに相当する「文字列・属性値」が得られる．前者はコマンドデコーダに送られ，描画コマンドのテンプレートが生成される．描画命令生成器は，これらのテンプレート，およびそれらと対応する文字列・属性値を受け取り，

実際の低位描画命令列を生成する．これらのデータの流れを制御する機構を具体化する．

(3) 上記の検討結果に基づき，①ビットマップ(bitmap)のみを用いる方法(BITMAP 方式)，②上述のように，実際に遷移を引き起こす入力の各ビットの共通の部分をマスク(mask)として記録し，マスクされないビット(offset)の上位数ビットについてのみビットマップを持つ方法(MASK/BITMAP 方式)，③遷移先状態がひとつのみの場合がしばしば現れることに着目し，そのような状態遷移についてのみ，入力を陽に比較できる仕組みを作り，上記の方法と組み合わせる手法(単遷移指向方式)，④前述の①，②における状態と入力の関係を逆転して遷移機構を実現する方法(入力ベース方式)，等について具体的にハードウェアを設計し，比較評価を行う．

(4) 一方，SVG 記述中の描画要素には，それらを並列に描画しても描画結果には影響を与えないものがある．ハードウェアにより効率良くデコードを行うためには，そのような並列描画可能な要素を抽出し，並列にデコード処理を行うことが重要である．逆に言えば，並列性のない要素は逐次的にデコードし，その順に描画する必要がある．与えられた SVG 記述中の描画要素の並列性を抽出するアルゴリズムを具体化するとともに，典型的な SVG 記述例を用いて得られる並列度を評価する．

(5) 上記の検討結果を基に，①バウンディッドボックス法に比べて高速に実行できるウィンドウ分割に基づく重なり判定アルゴリズム，②そのウィンドウサイズを可変化することによるアルゴリズムの最適化，③並列度を上げるために描画要素の分割を自動的に行うアルゴリズム，④描画要素の層間移動により，並列処理に必要となる描画ユニット数の削減手法,等をコーディングし,実際の SVG 記述を前処理できるソフトウェアツールを作成する．さらに，典型的な SVG 記述例を用いて得られる並列度を評価する．

４．研究成果

(1) 前節(3)に基づき，ストリームデータとして SVG テキストを入力すると，XML の構文解析(タグ・属性・内容の文字列への分割)，得られた文字列の SVG タグ・属性としてのマッチング解析，および，タグ・属性に対応する計算処理を行い，出力として OpenVG のコマンド・データを得る，SVG デコーダを設計した．構成を図２に示す．各構成要素は以下の処理を行う．

① 入力ユニット：SVG テキストをストリームデータとして入力し，1 文字を 7bit として構文解析ユニットへ送る．データは非同期で送受信する．

② 構文解析ユニット：テキストデータを１文字単位(7bit) で入力し，XML 構文解析を行い，タグ名文字列，属性名文字列，属性値文字列，内容文字列を取り出す．結果を，タグ名文字列はタグ名変換ユニットへ，属性名文字列は属性名変換ユニットへ，属性値文字列は属性値キューへ，内容文字列は内容キューへ，それぞれテキストデータとして１文字単位で出力する．

③ タグ名/属性名変換ユニット：テキストデータを１文字単位(7bit) で入力し，タグ名文字列/属性名文字列をタグ名 ID/属性名 ID に変換し，変換結果をキューへ出力する(8bit)．

④ キューユニット：以下の４つのデータを，FIFO のリスト構造で一時的に蓄える．

・ タグ名 ID（タグキュー）
・ 属性名 ID（属性名キュー）
・ 属性値文字列（属性値キュー）
・ 内容文字列（内容キュー）

４つのキューはそれぞれ独立している．タグ名 ID，属性名 ID は変換ユニットからエンキューする．属性値文字列，内容文字列は構文解析ユニットからエンキューする．すべてのキューは計算ユニットからの制御でデキューする．

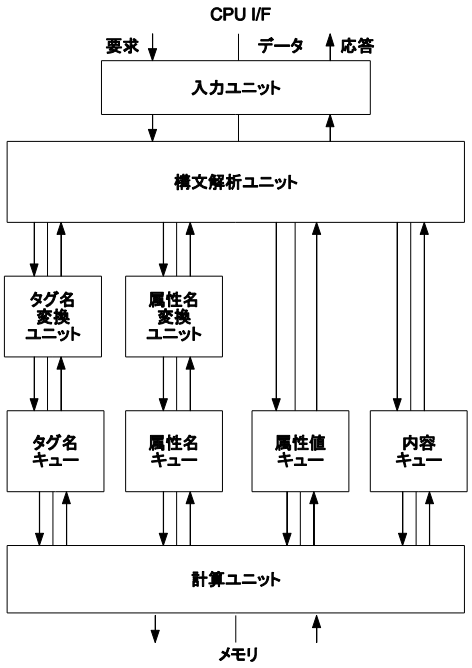

図２

⑤ 計算ユニット：キューからデータを受け取り，SVG を OpenVG の命令へ変換する．計算ユニットは，制御ブロック， 演算ブロック，キャッシュメモリから構成される．制御ブロックでは入力した，タグ ID，属性 ID，

属性値と内容に基づいて，計算命令を選択し，演算ブロックへ渡す．演算ブロックは制御ブロックからの命令に基づき，演算を行う．得られた OpenVG のコマンドとデータはキャッシュメモリに保存し，要素ごとに出力する．

(2) シミュレーションにより，SVG デコーダが必要とするメモリ量と OpenVG へのデコード時間を調べた．結果の一部を図３に示す．なお，BITMAP 方式，MASK/BITMAP 方式，単遷移指向方式，入力ベース方式をそれぞれ，B, M, S, I と表し，（構文解析ユニットの方式）/（タグ名変換ユニットの方式）の形で方式を示している．例えば，B(16)/M は，構文解析ユニットを 16 ビットの BITMAP 方式で構成し，タグ名変換ユニットを MASK/BITMAP 方式で構成することを表す．MASK/BITMAP 方式では，MASK 部 16 ビット，BITMAP 部 16 ビットの固定である．また，X はオートマトンの遷移に用いる内部メモリとデコード結果を格納する外部メモリのアクセス時間比で，例えば X=10 は，外部メモリのアクセス時間は，内部メモリの 10 倍であることを示す．

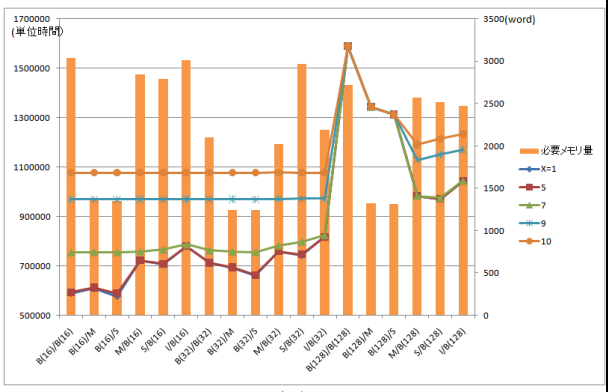

図３

この結果より，ビット長が 16 の BITMAP 方式が高速であるが，メモリの使用量が多い．また内部メモリと外部メモリのアクセス時間の差が大きくなるにつれ，全方式の差がなくなっていくのがわかる．しかし，ビット長が 128 の BITMAP 方式は BITMAP を読み込むための時間を多く要し，性能は良くない．メモリの 1 ワードを増やす必要がある．メモリの使用量という点では，単遷移指向方式や MASK/BITMAP 方式の効率が良いと言える．

(3) SVG 記述中の，並列に描画しても描画結果には影響を与えないものを抽出するツールに関して，以下のような検討結果を得た．
① 基本となる，ウィンドウ分割に基づく重なり判定アルゴリズムを開発した．これは，例えば図４のような図形に対し，ウィンドウ辺の中点で次々とウィンドウを分割し，図５のような解析木を構成することにより，並列に描画できる要素を抽出する．なお，図４において，数字は本来の描画順である．図５では，根から葉へ向かう親子関係が保存すべき描画順を表している．従って，例えば親子関係にない描画要素９と描画要素６は並列に描画できる．同一節にある要素は，その順に描く必要がある．例えば，描画要素８は描画要素２，４，６，７，９等と並列に描画できるが，１→３→５の後でなくてはならない．

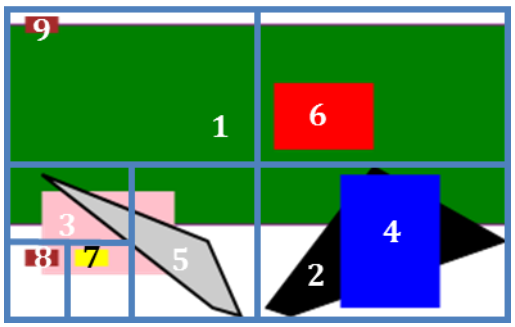

図４

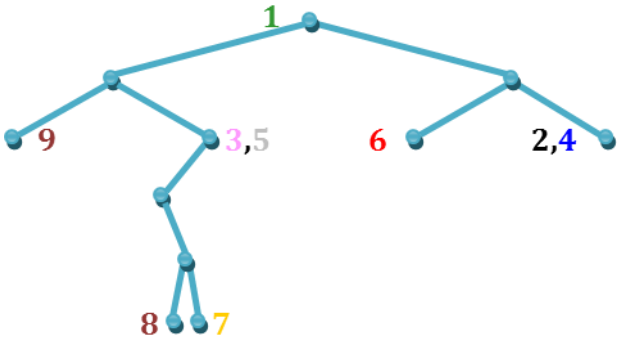

図５

② 上記アルゴリズムでは，例えば，図４のほぼ中央に小さな描画要素が存在する場合，実際には描画要素１以外とは並列に描画できるのにも関わらず，そのような並列性が抽出できない．そこで，ウィンドウの分割位置を，描画要素の位置に応じて適応的に決められるような，アダプティブなウィンドウ分割アルゴリズムを開発した．このアルゴリズムによるウィンドウ分割の様子を図６に示す．同図では，必ずしもウィンドウの中点でウィンドウが分割されていないことに注意する．

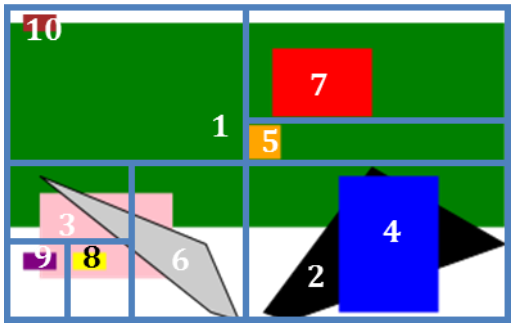

図６

③ さらに，ウィンドウの分割付近にある大きな描画要素を効率よく扱うために，描画要素の自動分割手法を開発した．ただし，この実装にはいくつかの注意点が必要であった．例えば，描画要素が非凸領域である場合，図７のように複数の描画点集合をつなぎ合わせて，新たな描画点集合を構築する必要がある．この自動分割により，分割限度を決める最小分割長を小さくしていくと，並列度を単調増加させることができる．この様子を図８示す．

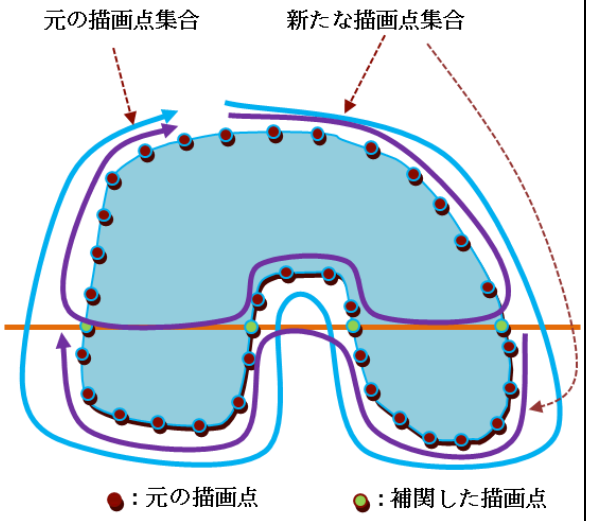

図７

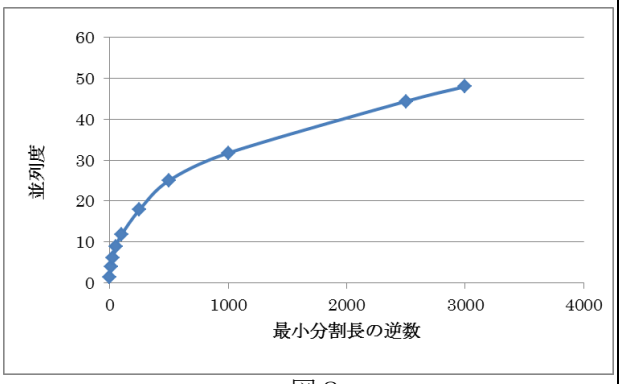

図８

④ 描画要素の分割により，並列度は上がるが，描画に必要となる描画ユニットも多数必要となる．実際に必要となる描画ユニット数を削減するために，描画要素の層間移動を行うアルゴズムを開発した．図９は，描画要素と描画ユニットの関係を表す．この図では，赤色の箱が各描画要素を表し，上に積み重ねられた描画要素は下の描画要素の後に描画する必要があることを表す．従って，この例では，最初の５個の描画要素が５つの描画ユニットを使って並列に描かれ，次に，その上に４つの描画要素が並列に描かれる．６回目の描画で最後の描画要素が描かれる．これを実現するには，５つの描画ユニットが必要となるが，それらすべてが必要となるのは最初のみである．ここで，必要な描画順序は縦関係で表されているため，この関係を保存していれば，描画を遅らせても描画結果には影響ないことに注意する．この考察に基づき，描画を遅らせた例を図１０に示す．この例では，左端およびその右の描画要素の描画を２回目から行い，４番目の描画要素の描画を５回目から行ったものである．その結果，同時に必要となる描画ユニット数を３に減らすことができている．このような描画要素の層間移動を実装した．その結果，描画要素の分割を行っても，必要となる描画ユニット数をある程度押さえることができた．図１１は，この層間移動を適用した際に，必要となる描画ユニット数が減らせる様子を示している．

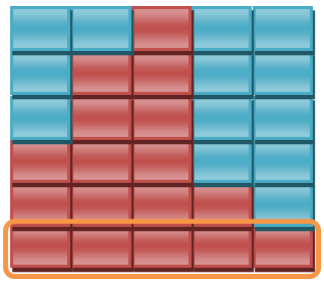

図９

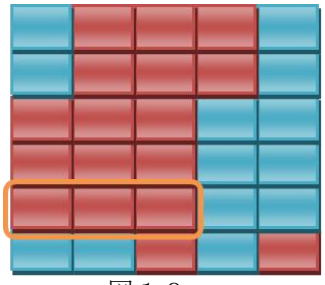

図１０

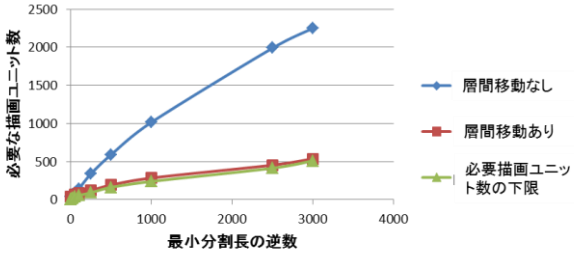

図１１

５．主な発表論文等
（研究代表者，研究分担者及び連携研究者には下線）

〔雑誌論文〕（計１件）

① C. Mannakkara, T. Yoneda, Asynchronous Pipeline Controller Based on Early Acknowledgement Protocol, 電子情報通信学会英文論文誌, 査読有, E93-D, No.8, pp.2145-2161

６．研究組織
(1)研究代表者

米田　友洋（YONEDA TOMOHIRO）
国立情報学研究所･アーキテクチャ科学研究系・教授
研究者番号：３０１８２８５１