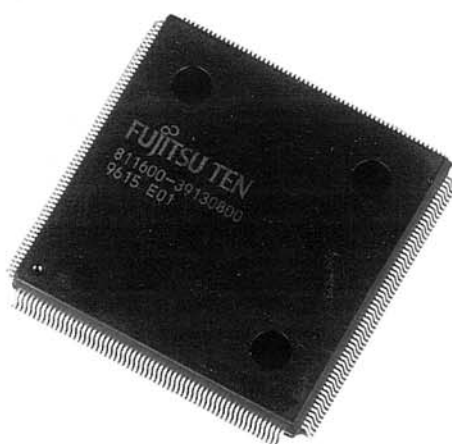


VICS用システムLSI

System LSI of VICS

田中 寿夫 *Toshio Tanaka*
谷口 功 *Isao Taniguchi*
伏見 文孝 *Fumitaka Fushimi*
前田 恵一 *Keiichi Maeda*



要 旨

VICS（道路交通情報通信システム）を搭載したAV機器において、分散している各機能を1チップに統合化したVICS用システムLSIの開発を行った。

開発では、従来、専用プロセッサと専用LSIで処理していたグラフィックス描画機能のソフトウェア化を図り、表示制御機能を中心にDRAM制御、割込み制御およびシリアル、パラレル、タイマなど各周辺機能をトータル55Kゲート規模、240ピンQFPパッケージに集積している。

高速化とローコスト化の両立は、VICS簡易図形に適した描画ソフトウェアの開発と、RISC32ビットCPUで実現した。また、ゲート規模拡大を抑えるため、外部バス幅を32ビットにすることでユニファイド・メモリ・アーキテクチャによるアクセス競合の調停制御を簡素化している。

Abstract

We have developed a system LSI for VICS(Vehicle Information & Communications System). Each individual function of audio and visual equipment has been integrated onto a single chip.

A graphics drawing function is accomplished by a CPU and its software which had been done by an individual ASIC and processor.

The system LSI includes display control, DRAM control, interrupt control, serial I/O, parallel I/O, timer function and so on. The gate scale is about 55k and its package is 240 pin QFP.

Compatibility between agility and lower cost was achieved with a newly developed drawing software for VICS and employment of a 32 bit RISC CPU.

Bus arbitration control under the unified memory architecture is fulfilled with adapting 32 bit external bus without gate scale expansion.

1. まえがき

1996年4月23日、都市部での渋滞を緩和するため、ドライバに必要な道路交通情報を提供する「VICS (Vehicle Information and Communication System)」が実用化された。^{脚注1} VICS情報は、文字、簡易図形、地図リンクの3つの形で提供される。簡易図形による情報提供は、情報の伝達性の点で車載用として適している。しかし、商品化に対し、簡易図形描画に適したLSIが存在しない。そのため、本稿では、独自にソフトウェアとハードウェアの最適化を図り、簡易図形描画に適したVICS用システムLSIについて発表する。

我々は、まず、システムの成立性を検証した。ここでは、「VICS公開デモ実験」での情報を用い、描画性能として目標値3秒以内を実現できる目処を立てた。

設計は、コスト低減を図るために試作段階では描画プロセッサ処理していた描画機能をソフトウェア化するとともに、グラフィックス専用メモリを無くし、メインメモリと併用するユニファイド・メモリ・アーキテクチャ方式（以下UMAと記す）を採用することにした。

一方、高速化のためにRISC32ビットCPUを採用し、外部メモリアクセスを32ビット幅で実現することとした。以上を達成するVICS用システムLSIの開発を行った。

本LSIを用いた商品開発は、自動車メーカ向けVICSユニットとして、1996年4月より量産を開始している。

2. システムの成立性

システム性能は、簡易図形の描画時間で決まる。よって、仕様決定は、描画性能をどこに設定するかがコストに大きく関わる重要なポイントである。

そのため、「VICS公開デモ実験」で用いた各種情報の中で最も時間がかかる「一般道広域」情報を用いて各種ハードウェアによる描画速度の調査を行った。表-1に描画時間比較表を示す。

こでの描画速度は、FM多重放送のレベル2データであるベクトルデータをグラフィックス表示データに変換する時間を示す。その結果、試作段階で約4.8秒かかっていた描画時間が32ビットパソコン (i486SX/16 MHz) で約2.3秒となる。しかし、本システムでは、UMAによる表示アクセスとの競合待ち、RTOSやFM多重／ビーコンの受信割り込みが同時発生する場合があるため、要求性能を3秒以下とし、32ビットバス化と、内部レジスタアク

表-1 描画時間比較表

	描画時間*	FM多重、ビーコン 受信割り込み処理	リアルタイム OS (RTOS)	表示制御と のアクセス競合
デモ実験機	4.8sec	○	×	×
パソコン (i486SX/16M)	2.3sec	×	×	×
要求仕様	3.0sec	○	○	○

*VICS公開デモ実験「一般道広域」画面

セスの高速化を図った方式で開発することとした。

3. LSI化の特徴

LSI化に際しては、①システムの最適化、②描画処理の高速化、③独自機能の搭載、3つの要求を満たすことを目標に開発している。

3.1 システムの最適化

システムの最適化は、①32ビットバス幅でのメモリアクセス化、②LSI内部レジスタアクセスの高速化、③描画アルゴリズムに適したハードウェア構成により実現している。

1) 32ビットバス幅でのメモリアクセス化

本システムでは、グラフィックス表示領域、FM多重データ領域、プログラムワーク領域を同一DRAM内にマッピングしている。このため、FM多重、プログラムワーク領域のアクセスとグラフィック表示アクセスが競合した場合、16ビットバス幅ではCPUアクセスが1秒当たり最大232 msec占有してしまう。(320ドット×240ドット×4画面を60画面/秒で繰り返しアクセスがあると、画像データとして18.5Mbit/秒となり、これを16bitバス幅でアクセスすることで1,156,000回のアクセス=232 msecとなる。)

このCPUのDRAMアクセス待ちにより、パフォーマンスが23%低下する。これを32ビット化することで、1/2に抑制できる。また、32ビット化によりキャッシュを設けることなく表示アクセスとCPUアクセスのアービトレーション機能を実現している。図-1にブロック図を示す。

2) LSI内部レジスタアクセスの高速化

LSIの内部レジスタは、58種類ある。この中で、パレットレジスタやシリアル通信用レジスタは、頻繁にアクセスが発生する。そこで、本システムでは、LSI内部レ

脚注1：3メディア対応VICS車載34頁機参照

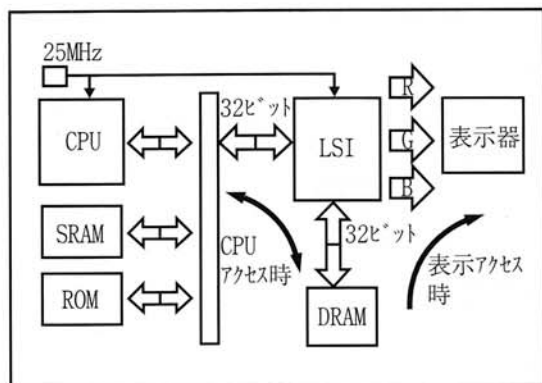


図-1 32ビットバス幅でのメモリアクセス化
Fig.1 Memory access with 32 bit bus

ジスタアクセス時間を3サイクル (120 nsec) に設定し、他の領域 (フラッシュメモリ、マスクROM, SRAM, DRAM各領域) より高速化することでCPU 動作の待ち時間を低減している (図-2)。表-2に各メモリ、レジスタ領域のアクセス時間を示す。

表-2 各メモリ/レジスタアクセスサイクル

	サイクル数	アクセス時間	アクセスバス幅
ROM 領域 (マスク)	5	200nsec	16/32ビット
	4	160nsec	16/32ビット
漢字フォント領域	6	240nsec	8/16/32ビット
SRAM領域	4	160nsec	8/16/32ビット
内部レジスタ領域	3	120nsec	32ビット
DRAM領域	5	200nsec	8/16/32ビット

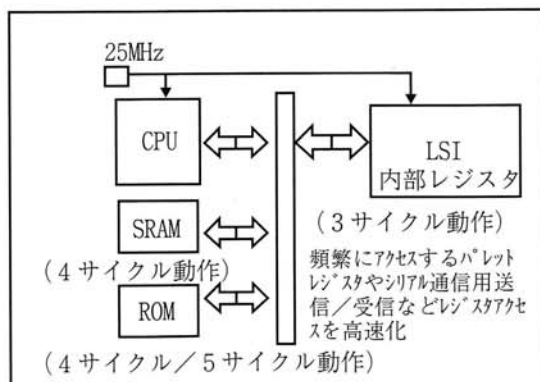


図-2 LSI内部レジスタアクセスの高速化
Fig.2 Improvement of internal register access speed

3) 描画アルゴリズムに適したハードウェア構成

FM多重の簡易図形表示は、区分地図や道路を模式化した固定的な下絵データ部と、渋滞区間や事故発生場所等を表すような可変的な上書きデータ部に分かれている。情報の表示は、この下絵図形に上書き図形を重畳して行う。下絵データは、時間的に変化しない情報をもっているため、上書きデータより大容量となる。よって仮想画面を16ブロック準備しておき、複雑な下絵データを予め、描画変換 (ベクトルデータからマップデータへ変換) しておき、表示要求があった場合、データ量の少ない上書きデータのみを変換することで、擬似的に高速化を図っている。図-3にそのマッピング構成を示す。

また、水平、垂直、クロスハッチング処理では、ワーク領域に図形とハッチング図形とハッチング模様を論理的な処理で生成できるようにアドレスマッピングしており、複雑な図形にハッチング模様のマッピングが高速かつ簡易なアルゴリズムで可能としている。図-4にクロスハッチング処理のアルゴリズムを示す。

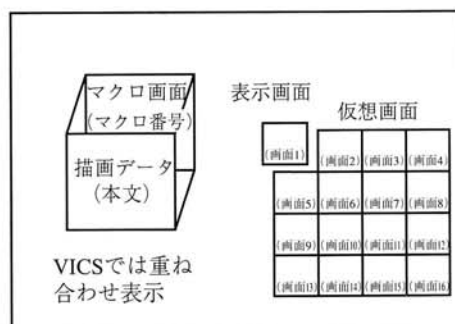


図-3 画面のマッピング構成
Fig.3 Mapping structure of a picture

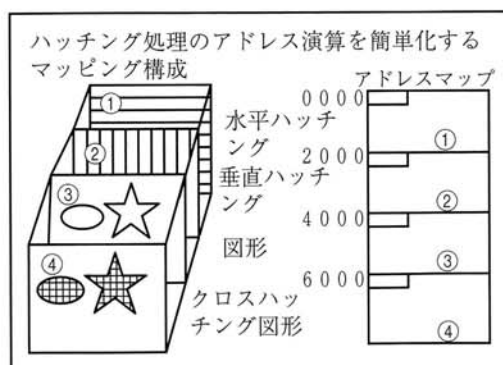


図-4 クロスハッチング処理方式
Fig.4 Cross hatching process

3. 2 描画処理の高速化

本LSIは周辺回路のシンプル化のために、専用プロセッサと専用LSIで処理していた描画機能を、ソフトウェア化している。そのため、以下のことに重点を置き描画の高速化・ソフトウェア化を行った。

- ①描画面面へのアクセス回数の削減
- ②描画アルゴリズムの最適化

3. 2. 1 32ビット一括処理

点を打つのはグラフィックスの基本である。その点を打つために必要な情報は、点の座標位置と色である。この点1個さえ打てれば、直線も描画できるし円も描画できる。1点を描画するまでの大まかな流れとイメージ図を示すと図-5のようになる。

しかし、図-5に示す様な「8ビットで1画面アクセス」では点の色までを考慮した場合、高速な表示は期待できない。一つの表示画面は4枚の画面で構成されるため、

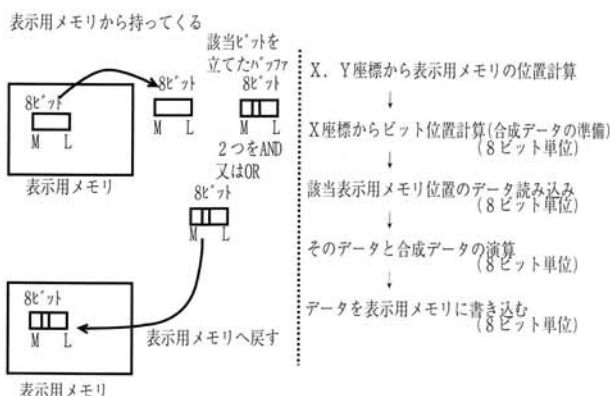


図-5 1点(ドット)描画イメージと流れ-改善前

Fig.5 Flow of 1 dot drawing (conventional)

4画面分アクセスする必要がある。そこで、32ビット単位でのオペレーションを考え、「32ビットで2画面同時アクセス」を行っている。つまり、32ビットを上位・下位の半分に分けそれぞれで別の画面にアクセスする様にしている。この方法で表示画面に1点を表示するのに、2画面分のアクセスで済んだ。(図-6)

3. 2. 2 描画アルゴリズム

描画機能のソフトウェア化を行う際に、VICSで提供される文字や図形に対応する必要があった。レベル2(簡易図形表示)では、表示処理の負荷が大きい情報画面もあった。そこで以下のことに注意して取り組んだ。

1) 整数演算処理

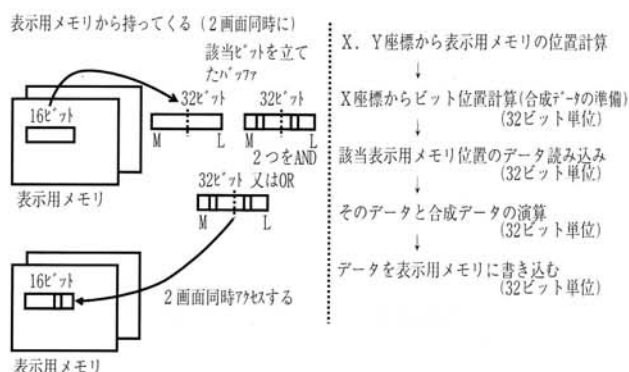


図-6 1点(1ドット)描画イメージと流れ-改善

Fig.6 Flow of 1 dot drawing (improved)

直線や円等の図形を描画する場合、その手法はさまざま考えられる。例えば、円であれば数学的に円の2次方程式(中心座標: (x_c, y_c) 、半径: r)、

$$(x - x_c)^2 + (y - y_c)^2 = r^2$$

から y について解いた

$$y = y_c \pm \sqrt{r^2 - (x - x_c)^2}$$

もしくは極座標表現をした

$$x = x_c + r \cos \theta, \quad y = y_c + r \sin \theta$$

(但し、 $0 \leq \theta < 2\pi$)

を使用すれば描画はできる。しかし、これらを使用する場合、演算処理において浮動小数点演算を必要とし高速性をそこなう。そこで、直線描画にはDDA(デジタル微分解析: Digital Differential Analyzer)手法を、円及び楕円描画にはブレンハムの円弧発生方法(Bresenham's Circle Algorithm)を使用した。これらの手法は共に、整数演算のみで図形を描画するため高速である。^{4), 5)}

2) 塗り潰し処理

矩形や円等の塗り潰し処理を行う場合も、手法はさまざま考えられる。例えば、塗り潰す図形内部の1点を求める。それを中心にして、1点(1ドット)ずつ図形内部を塗り潰していけば描画できる。しかし、図形によっては与えられた情報(座標位置)から外観が判る場合もある(矩形、円等)。この場合は、前出のような塗り潰し処理では高速な描画は期待できない。しかし、水平方向へは塗り潰す範囲が判っていれば、数ドットをまとめて描画する(最大16ドット)ことで、高速な描画を実現できる。また、垂直方向へも、単純な整数演算処理で行えるので高速な描画が実現できる。これらを利用して高速な図形の塗り潰し処理を実現した。

以上の取り組みの統合により、表示処理負荷の大きい

レベル2（簡易図形表示）の画面を表示するまでの時間として、約2.5秒を達成した。

3.3 独自機能の搭載

3.3.1 パレットレジスタ書き換え時の画面乱れ低減機能

本LSIのパレットレジスタは、2段レジスタ構成を採っている。

従来の1段構成だと、CPUがパレットデータを更新するタイミングが、表示期間中に行われると表示画面乱れを発生するため、更新タイミングに大きな制約が生じる。

しかし、2段構成にすることで、CPUは、1段目のレジスタにデータを更新し、LSIが画面切換え時に2段目のレジスタにコピーするので、CPUの更新タイミングに対する制約を大幅に低減している。(図-7 参照)

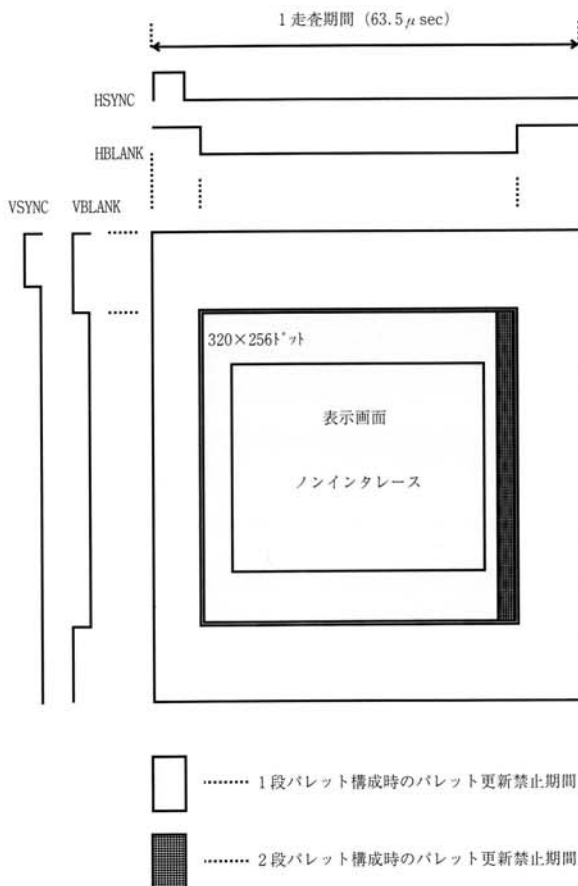


図-7 パレット更新禁止区間の比較

Fig.7 Relaxed timing restrictions for rewriting

3.3.2 外部同期機能

本LSIは、映像信号（RGB）をドットクロックに同期し送出する。

そして、この映像信号を、LSI内部で生成した水平、垂直同期信号（HSYNC, VSYNC）のエッジタイミングで送出する内部同期モードと、LSI外部からの水平、垂直同期信号のエッジタイミングで送出する外部同期モードがある。

本LSIでは、この外部同期モードにより、VICS-LSI単体での表示だけでなく、他表示ユニットとディスプレイの共有を可能にしている。

また、外部同期モードの場合、他表示ユニット描画部と、VICS-LSI描画部が同一画面に共存するが、外部同期信号の遅延等のばらつきで、表示開始位置がずれることに備え、ソフトにより、標準値より水平開始位置（+10～-5ドット）と垂直開始位置（±3水平）を可変にし、表示画面の微調整を可能にしている。

4. VICSユニットへの応用

VICSユニットは、FMチューナ部、多重デコーダ部、通信処理部、データ処理部、簡易図形描画部から構成される。この中で通信処理、データ処理、簡易図形描画部をLSI化している。図-8にそのブロック図を示す。

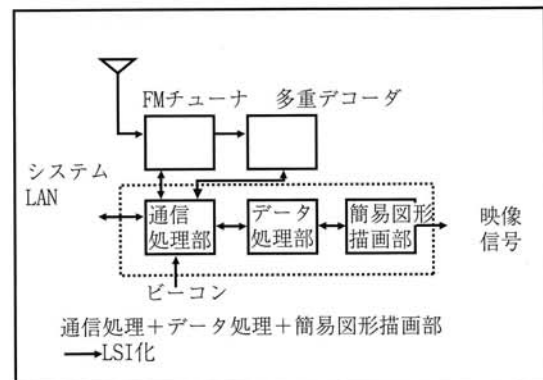


図-8 VICSユニットの内部ブロック図

Fig.8 Block diagram of VICS ECU

本LSIにより、部品（主要LSI：7個、ロジックIC：20個程度）を削減できた。図-9に従来品のブロック図、図-10に開発品のブロック図を示す。LSIの実現は、仕様変更や開発期間の点で有利なCMOSゲートアレイを用いている。

本LSIを使用することにより、速度的には、約2倍、コスト的には、40～60%ダウン、部品点数25点→1点、消費電力1/5を達成できた。表-3にLSI化による効果を示す。

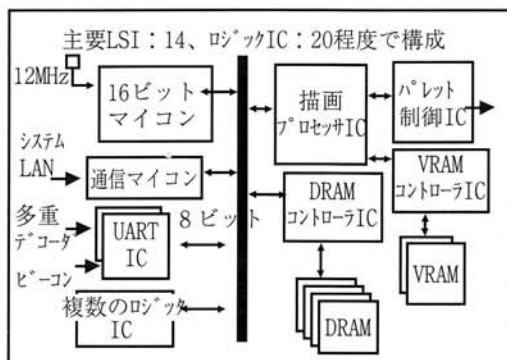


図-9 従来の構成

Fig.9 Block diagram (conventional) 20 ICs are included

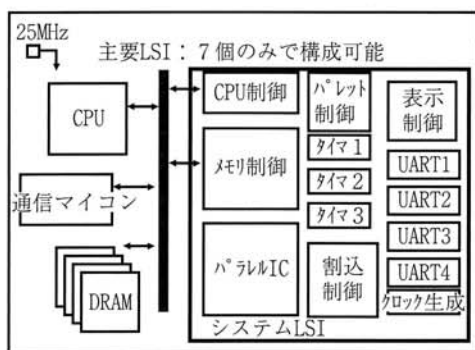


図-10 システムLSI化による構成

Fig.10 Block diagram with system LSI reduced to 7 ICs

5. まとめ

VICS簡易図形処理に適したシステム設計を実施し、LSI化を行うことで目標の描画速度を達成できた。今後、システム化の核となるCPU機能を含めたLSI化の取り組み、システムLSI化するアーキテクチャの成立性の検証手法を確立していきたい。そして、セミカスタムLSIの強みである短期開発に対応できるようなシステム評価技術の向上を図りたい。

[参考文献]

- 1) 田中寿夫, 山本徹二, 福田晋児: "VICS車載装置", 富士通テン技報, VOL12, NO1, pp.41-48(1994-06)
- 2) 佐々木三利, 佐々木満, 井手賢一郎: "移動体受信用FM多重放送受信機", 富士通テン技報, VOL12, NO1, pp.49-54(1994-06)

表-3 LSI化の効果

	効果
部品点数	25点 → 1点
コストダウン	40% ~ 60%
消費電力	約 1/5

- 3) 新宮威一, "自動車交通情報・運転システムの研究開発動向", 富士通テン技報, VOL11, NO2, pp.1-14(1993-07)
- 4) S. ハリントン: アルゴリズムとプログラムによるコンピュータグラフィックス [1], マグロウヒル出版(1984)
- 5) 山口富士夫 監修: 実践コンピュータグラフィックス, 日刊工業新聞社(1987)

筆者紹介



田中 寿夫（たなか としお）

1980年入社。以来画像符号化の応用研究、自動車関連通信機器の開発に従事。
現在LSI開発部企画・開発プロジェクト課在籍。



谷口 功（たにぐち いさお）

1993年入社。以来新分野商品の制御ソフトの開発に従事後、デジタルICの設計・開発に従事。
現在LSI開発部設計プロジェクト課在籍。



伏見 文孝（ふしみ ふみたか）

1993年入社。以来カーオーディオの要素開発に従事後、デジタルICの設計・開発に従事。
現在LSI開発部設計プロジェクト課在籍。



前田 恵一（まえだ けいいち）

1974年入社。以来無線機関連の開発を経て、1995年よりICの開発に従事。
現在LSI開発部次長。

