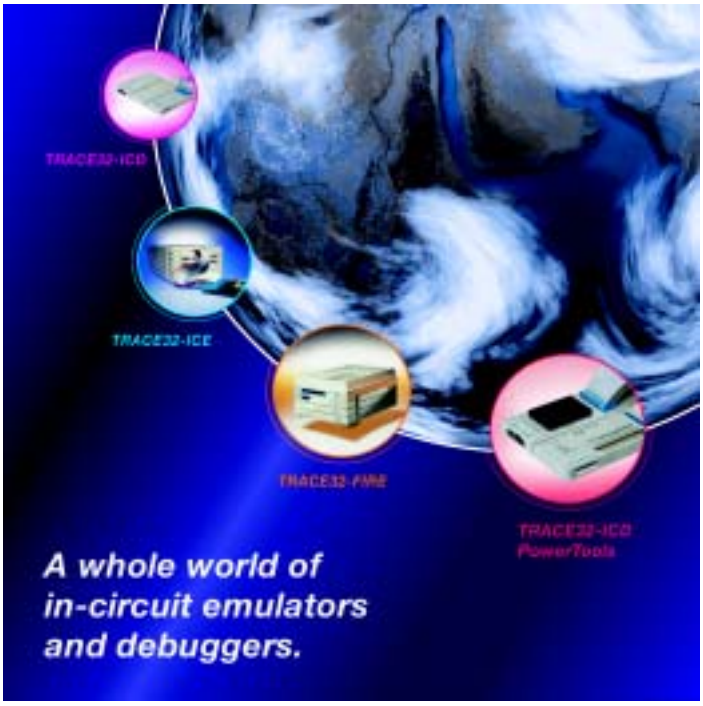
NEWS

TRACE32®
IN-CIRCUIT EMULATORS & DEBUGGERS
ONE SYSTEM FITS ALL!

NEWS FOR THE EMBEDDED SYSTEMS CONFERENCE

セールスの増加

ハイテク分野の企業にとって大変厳しかった2002年度、ローターバッハは売上高20%アップという大きな成功を収めることができました。2002年の経済環境の中で成長を実現したことはハイテク市場、とりわけエンベデッドツール業界では特筆すべきこととして注目されています。昨年度は、ハイテク業界の多くの企業にとって試練の年でした。また、組込み開発ツールの市場も成長したとは考えにくい状況でした。つまり、ローターバッハはマーケット全体においてより多くのシェアを獲得したことになります。言い換えれば、マーケットにおける私どもの地盤がより強固なものとなったということになります。このことは英国に新設された現地法人が予定通りの売上を達成、初年度から黒字転換したことからも見て取れます。
2003 年度が成功の年になるかどうかはもちろん、誰も予想することはできませんが、ローターバッハは極めて良い状態で新年度をスタートすることができました。



日本法人を新設

日本はご存知のように、ハイテク分野では重要なマーケットです。また同時に、外国企業にとって、容易に参入できるマーケットではないこともよく知られています。とりわけ、組込み開発ツールの分野では、市場の要求に対応した製品をハイレベルな技術サポートと合わせて提供できることが重要です。このためには現地で直接活動することが、より良い参入方法であると考え、2003 年 2 月に日本法人を設立しました。2003 年度は日本のお客様にローターバッハ製品を十分ご活用頂ける販売およびサポート体制を構築して参ります。
このことは、私どもの国際的プレゼンスをさらに強化する上での重要なステップとなります。

開発チームの増強

2002 年度の終わりから 2003 年の初めにかけて、2003 年度に計画されている全てのプロジェクトを迅速、かつ、スケジュール通りに遂行するため本社 Hofolding の人員を増強、TRACE32 製品の VHDL 設計及び JTAG ツールの開発体制を一段と強化しました。

新製品

次ページ以降をご覧に頂くとおわかりの通り、ローターバッハには数多くの新製品と構想があります。このなかで中核となるのは、OMAP 等のサポートに不可欠なマルチコアデバッグと DSP 用デバッガのサポート領域の拡大です。詳細につきましては、最寄の支店、又は代理店にお問合せください。

LAUTERBACH

マルチコアデバッグ

複数のコアを持つシステムオンチップデザインのデバッグが、開発ツールに新たな要件を提示しています。理想から言えば、チップデザイナーとツール製作者は、ユーザーが自由なコアの組合せに対応する高品質の開発ツールを、素早く入手できることを保証する適切な技術について早急に合意すべきでしょう。

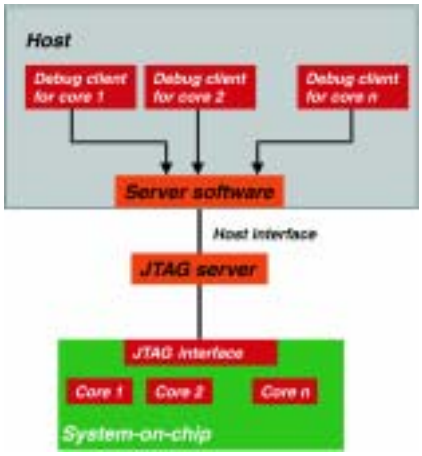
ASIC は現在、組込み開発でその応用範囲が広がりつつあります。最適な機能性と性能を達成するために、SoC 形成に複数のコアを統合することが一般的になっており、中でも DSP と組み合わせた RISC プロセッサの使用が広がっています。
具体例としてはテキサスインスツルメントの OMAP とインフィニオンの S-Gold があります：
・OMAP は ARM925/926RISC コアと TMS320C55ｘDSP が統合されたプロセッサです。

・S-GOLD は ARM926EJ-S と２つの OAK DSP が統合されたプロセッサです。

この種のマルチコア SoC デザインの統合化およびテストフェーズでは、マイクロプロセッサ開発ツールに対し新たな機能が要求されます。ここでは、共通のデバッグインタフェースを通じて複数のコアをアドレッシングして行うデバッグ手法に焦点をあて解説します。
まず、幾つかのアプローチを示し、それぞれの問題点を説明します。

一つのデバッグインタフェースを通じて複数のコアをアドレッシング

近年、多くのプロセッサがオンチップデバッグ回路を備えています。デバッグ回路をコントロールするために最も頻繁に使われるインタフェースは、JTAG の仕様によって定義されています。しかし、マルチコア SoC デザインでは、ピン数の節約およびコストの削減のため、すべてのコア用にそれぞれの JTAG インタフェースは用意されてはいません。代わりに、ジョイント JTAG インタフェースにて全てのコアのデバッグに対応しなければなりません。
この場合、デバッガがジョイント JTAG インタフェースを通じて複数のコアを特定する方法および同期デバックを行う方法が疑問点として提起されます。



１．１つのデバッガで SoC 内の全てのコアをサポート

この方法は一見して開発者にとって理想的なソリューションに思えます。この方法の主な優位性は、全てのコアに適用することができる開発ツールであるため、一つのユーザーインタフェースと製品の仕組みを習得するだけでよいということです。これは、大量生産され、数多くの開発プロジェクトに使用される SoC を採用する場合では、品質の高いツールの入手が可能になり、確かに最良のアプローチといえます。このようなソリューションの一例として、ARM925/926 コアと同様に TM320C55xSDP のデバッグ可能とする、ローターバッハの OMAP デバッガがあります。

その一方、たったひとつの製品の開発のために特化した SoC を採用する市場の傾向もあります。次期製品では、パフォーマンスと機能性がより最適化された新しい SoC がさらに作り出されたりします。開発者が統合化されるコアを自由に選択したいと思えば、採用されるすべてのアーキテクチャに対応した最適な製品を提供することが可能なツール製作者のサービスもまた必要となります。当然、最初のチップがウエハーから製造された時点までに、デバッグツールも完全にテストされ利用可能状態となっていなければなりません。
したがって、この解決方法は明らかに最良な方法ではありますが、どのような SoC に対しても高い品質基準を維持しながらより迅速に、より安価に実現するのは大変困難なことです。

2．JTAG サーバーの使用

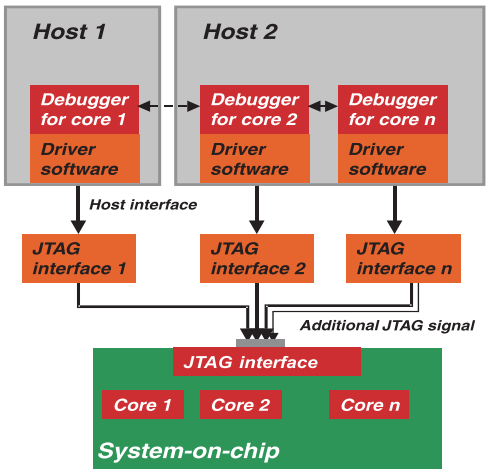
現在すでに、多くのコア用に完成度の高いデバッガが利用可能です。したがって、ジョイント JTAG インタフェースを介し、各コアのアドレッシングを管理する JTAG サーバーと既存のデバッガを接続する方法が考えられます。

このソリューションは次のようにして実現することが考えられます：インタフェースハードウェアはジョイント JTAG インタフェースの手前で接続され、ハードウェアはホスト上のサーバソフトウェアによって制御されます。今、それぞれのデバッグクライアントは、リモート関数コールを使って、サーバーにコミュニケーションリクエストを送信します。これにより、ジョイント JTAG インタフェースへの排他的なアクセスを保証、各コアに対して通信リクエストを送信します。図１にこのサーバーソリューションの概要を示します。

ここでは、デバッガは JTAG サーバーとのインタフェースに、専用の JTAG ハードウェアの代わりにソフトウェアインタフェースを使用しています。このアプローチは一見する、とてもすっきりしているように見えますが、先と同様に次の理由から、問題に対する完全な解決策とはなっていません。既に多くの半導体ベンダーから、それぞれの製品レンジの中でコアグルーピングを行うための JTAG サーバー製品が提供されていますが、各ベンダ間で中立的な見地に立ったサーバ基準はまだ存在しません。

JTAG サーバーによるソリューションはまた、各半導体ベンダがデバッグ機能の追加のためにそれぞれのコアの JTAG 信号を拡張すると、むしろ柔軟性が損なわれてしまいます。例えば、マルチコア開発環境では、それぞれのコアが直ちにコアの停止を可能にするストップリクエスト信号を持つことが望まれます。一方、停止表示信号によって、コアが停止したことを示すことも必要になるでしょう。　マルチコアアプリケーションで、全てのコアを同期して停止するために両方の信号が使用できれば理想的と言えます。

最適なコアの組合せを自由に選択する場合、かなりの追加信号がすぐに出てきます。簡易的な選択として、単純にこれらの信号はボード上を通過さ



せ、その追加機能を省いてしまうことでしょう。しかし、これらの信号は頻繁に必要であったり、少なくとも大変役立つため、機能させる必要があります。このため、JTAG サーバーでも特定アプリケーションに適合させ、また然るべくテストされなければならなくなるでしょう。

もう一つの重要な点は、特にリアルタイムアプリケーションにおける JTAG サーバーが取り扱うべき膨大なデータ量です。各コアの応答時間はその量に応じて遅くなります。したがってシステムのリアルタイム要件を満たすために、タイムクリティカル機能をハードウェアに盛り込む必要があります。　この場合もやはり JTAG サーバーを特定のアプリケーションに適合させることになります。

以上より、JTAG サーバーは一つの SoC 上で使用される全てのコアが同一の半導体メーカーから供給され、その半導体メーカーが JTAG サーバーを提供し、そのサーバーに適合するデバッガを同時に供給される場合にのみ最適なソリューションと言えます。一方、特定アプリケーション向けサーバを実現するには、デバッガ製作者間の協定が必要となります。通常、競合状態にあるツールベンダー間において、このような協定を取り交わすのは困難であり、時間的浪費やコストアップにもなります。更には、新しくコアを組み合わせる度に、JTAG サーバーをそれに応じて適応させなければなりません。

3．ジョイント JTAG インタフェースでの完全独立型デバッガ

このアプローチでは、シンプルでかつオープンな解決方法を示しています。この方法では、可能な

限りの最良のコアの組み合わせを、特定のアプリケーションに対応できるように自由に選択することができます。
開発工程の全ての段階を通じて、開発環境を特定のアプリケーションに適応させる必要はなく、完全に最適化されたデバッガを利用することができます。
ここでの基本的な考え方は、独立したデバッガがそれぞれのコア用のジョイント JTAG インタフェースに接続するというものです。このアプローチを機能させるためには、各デバッガは、対応するコアとデータをやり取りしていない時は、その JTAG ドライバを無効にしておく必要があります。３ページの図２に、このアプローチを示します。

今、同じ JTAG インタフェースを使用する複数の独立したデバッガがあるため、何時でもひとつのデバッガのみがインタフェースで動作することを保証する必要があります。　これは、ホスト上のデバッグタスクがセマフォシステムを使用し、JTAG インタフェースへの排他的アクセス権をどのデバッガに与えるか定義することにより、自動的に保証されます。もう一つの方法として、デバッガ間でのハードウェアセマフォを使用することも考えられます。

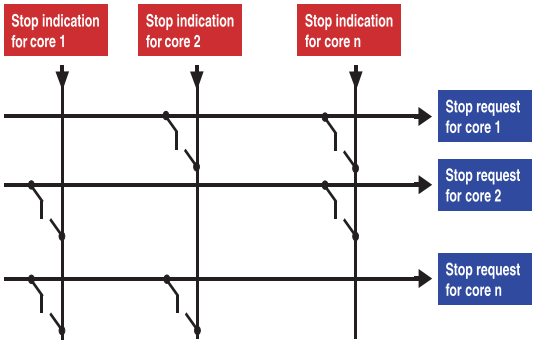
全てのコアのスタートおよびストップの同期方法に関する問題は、SoC 上のスペシャルロジックを通じて簡単にかつ効果的に解決することができます。例えば、次の２つの信号を追加することにより、すべてのコアを同時に停止させることができます。

・STOP Request Signal --- CPU コアを即座に停止させることができる信号.

・STOP Indication Signal --- CPU コアが停止したことを示す信号.

これらの信号は、SoC 上にマトリックスにて結合されているとします。これにより、例えばメモリマップドコントローラレジスタの設定にて、各コアは他のコアがストップした場合に、停止したいか、実行を続けたいかセットすることができます。図３はこのマトリックス例を示しています。　またこのマトリックスは、個々のコアが持つペリフェラルを含めることができるように簡単に拡張できます。

このソリューションの主な利点は、ストップ動作時の高度な同期性能とデバッガ製造者のための標準インタフェースがチップ設計者から提供されることです。　チップ設計者とツール製作者にとっては、このソリューションは比較的簡単手順で実



現できます。　２，３の基本的な前提条件がわかれば、どのようなコアでも SoC 上に統合することができ、対応する開発環境を自由に選ぶことができます。

ここでの利点は、次のようにまとめることができます。

・このアプローチでは、ツールベンダー間で取り交わされる追加協定は必要ありません。
高性能開発ツールをすぐに利用できます。

・ここで述べてきたソリューションでは、ツール製作者による特定アプリケーション向けの修正は必要ありません。他の開発プロジェクトでも再利用可能な標準的なツールを採用することができます。

・すべてのコアに対応する完全なデバッグシステムを得なければならないということは一見してマイナス要因に見えますが、よく見ると結局利点の方が多いことがわかります。通常、開発段階では、一度にテストされるのは一つのコアのみで、デバッガを個別に使用することができます。全てのデバッガを同時に使用しなければならないのは、コア間のインタアクションをテストする統合段階においてのみです。

・ローターバッハは、すでに数多くのコアを標準ユーザーインタフェースと共通の製品設計思想にてサポートしています。したがって、マルチコア SoC 設計用デバッガは既に存在しているといえます。

マルチコアデバッグが初期段階にあり、共通規格が定義されない限り、ローターバッハはこの三番目の解決策が最も効果的であると考えます。ローターバッハのデバッガは、ジョイント JTAG インタフェースでの動作のすべての要求に合致するように設計されています。

ローターバッハツールによるマルチコアデバッグ

1．必要なデバッガハードウェア

複数のローターバッハデバッガをジョイント JTAG インタフェースに接続、いづれかの標準インタフェース（LPT,USB,または Ethernet)にてホストシステムと接続することにより、複数のコアを簡単にデバッグすることができます。各デバッガはそれぞれ各々の JTAG ケーブルを持っています。JTAG ケーブルの接続には二つの方法があります。

・ターゲットシステムに各 JTAG ケーブル用のコネクタがそれぞれ用意されている場合。

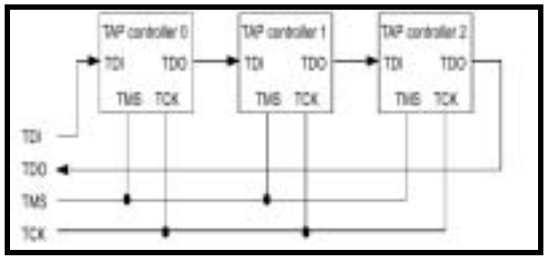
・ターゲットシステム上に存在する JTAG コネクタは一つのみ。この場合、すべての JTAG ケーブルのためのコネクタを持つ適切なアダプタが必要になります。図１を参照。

また、それぞれのコア用対応したトレースモジュールを追加することももちろん可能です。

2．コンフィギュレーション

ローターバッハのデバッガを構成する際に、注意しなければならない点が幾つかあります。
構成上の重要な要素は、コアが SoC 上でどのように配置されているかということです。
以下、個々のコアの持つ JTAG ポートが直列（デイジーチェーン）に配置された、単純かつよく使用される構成例をもとに説明します。図２参照。

最初の注意点は、各デバッガが対応するコアの特定方法です。JTAG の取り決めによると、個々の TAP コントローラがニュートラルに動作(BYPASS)することを保証するスキャンチェーン用シーケンスを生成するためのオプションがあります。従って、特定のコアをアドレスするには、スキャンチェーンを生成する際、対象となるコアの前後に、全て



の TAP コントローラ用の BYPASS シーケンスを生成するように、デバッガを構成しなければなりません。（図３上の IRPOST,IRPRE,DRPORT,および DRPRE を参照）

第２の注意点は、JTAG ポートへのアクセスが必要ない場合の個々のデバッガの動作です。
ジョイント JTAG インタフェースを使用していない時は何れのデバッガも、そのラインを Tristate モードに切り替えなければなりません。　それから JTAG ラインが次に有効になった時に、デバッガが定義された点からリスタートできるように、TAP コントローラ(TAP State)に対し適したステートを選択します。デバッガは、Tristate モードが有効になる前に、TAP コントローラをこのステートにセットし、JTAG ラインが再度有効となる時、TAP コントローラがこの状態になるようにします。

最も重要な JTAG ラインは、Tristate モードにおいて開放されたままのラインがないように、また、TAP コントローラが不用意に再構成されないように、抵抗でターミネーションされていなければなりません。構成作業を完了するためには、どのデバッガが SoC のリセットライン上でマスターとなるのか定義する必要があります。

Differential loading

ローターバッハは大容量プログラムをロードするための新手法を展開しています。これは、実際に変更された部分だけを開発ツールにロードすることを可能にするものです。通常、実際に変更されるのはプログラム全体の一部だけですので、ダウンロード時間を平均して

ほぼ３０分の１に短縮することができます。

一般的なソフトウェア開発は、デバッグ、エラーの修正、再コンパイル、再ロード、そしてまたデバッグというサイクルから構成されます。オンチップデバッグインタフェースのアーキテクチャ上、高速ローディングが難しいプロセッサにて大容量プログラムをテストする場合、多大の待ち時間を費やさなければなりません。

ローターバッハは現在、実際に変更されたプログラムの部分だけを新たにロードするという、ソフトウェアロード中の待ち時間を最小化するための機能を提供しています。
Differential Loading 機能の仕組みは以下の通りです。Load コマンドを入力する際、オプション機能である Differential Loading を有効にします。デバッガのローディングソフトウェアは変更されたプログラム部分を特定、それらを圧縮し、圧縮データをオンチップデバッグインタフェースを介し既に定義済みのメモリエリアにロードします。ローディングソフトウェアは次に、データを解凍するターゲットエージェントを起動、正しいメモリエリアに書き込まれていることを確認します。

Differential Loading の最初のサポートアーキテクチャは次の通りです。

・PPC7XX

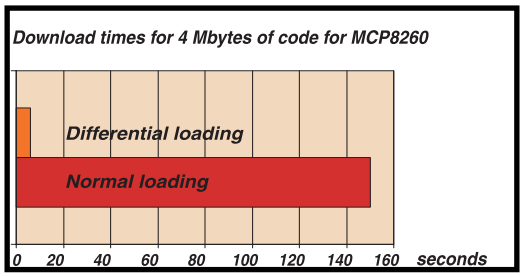
・MPC82XX

・JANUS

・ ARM with RTCK

下記 Web にて Differential Loading に関するより詳しい情報を参照できます。
http://www.lauterbach.com/lightspeed.html

ソフトウェアライセンスキー



ソフトウェアライセンスキーの認証を容易にするために、2002 年 12 月以降の全ての TRACE32-ICD には、デバッグケーブル上にライセンスキーが直接埋め込まれています。ライセンスファイルへのアクセスができないノートブック上あるいはテスト環境においても、問題なく個々のデバッグケーブルを使用することができます。

新リアルタイムカーネルサポート

・**QNX from QNX Systems Ltd.**
for ARM, PowerPC and XScale

・**µCLINUX**
for 68k, ARM and ColdFire

・**eCOS from Red Hat**
for PowerPC

・**Remote Debugging for OSE from OSE Systems**
for ARM

・**LINUX**
for ARM and XScale

TRACE32-PowerView でサポートされている全てのリアルタイムカーネルのリストは、弊社 Web 上で確認することができます。
http://www.lauterbach.com/xlist.html

THE PRODUCT LINE

TRACE32-ICD

New Debuggers

2003 年度、ローターバッハはインサーキットデバッガ TRACE32-ICD のサポートプロセッサを新たに追加します。最新 ARM コアと新 PowerPC ファミリはもとより、マルチコアデザインに主に使用される DSP を今年度はメインにフォーカスし、サポートしていきます。

ARM	ARM9EJ ARM10 Core ARM11 Core	now Q1/2003 Q3/2003
Avalent	Janus 2	now
DSP Group	TeakLite/OAK	Q2/2003
Hitachi	H8S2319, H8S2329, H8S2339	now
Infineon	S-GOLD (ARM926EJ-S and OAK)	now
Motorola	MPC5500 MGT5100 MGT5200 MPC750FX/CX/CXE MPC74xx MPC8265/MPC8264 MPC8540/MPC8560 MPC827x MPC828x MCS12C32 including on-chip trace	Q3/2003 now Q2/2003 Q1/2003 Q1/2003 Q1/2003 Q2/2003 Q2/2003 Q2/2003 now
STMicro-electronics	Super10	now
Texas Instruments	OMAP (ARM925/926 and TMS320C55x)	now

ARM 対応 ROM モニタ

2002 年 11 月から ARM 及び XScale 用の ROM モニタをご提供しています。ROM モニタは FLASH シミュレータ（ROM エミュレータ）と組み合わせ、若しくはシリアルインタフェース経由にて動作します。したがって、JTAG インタフェース無しで全ての ARM コアに対し TRACE32-ICD デバッガの使用が可能になりました。　例えば、ARM8 や StrongARM にも TRACE32-ICD デバッガが使用できます。

シリアル ROM monitor のライセンス

ホストのイーサネットアドレスでライセンスする現在のシステムに比べ、より柔軟性の高い USB ドングルが、シリアル ROM モニタのライセンス用に 2002 年 11 月より利用可能となりました。

FLASH ROM 内にソフトウェアブレークポイントを設定

TRACE32-ICD と PowerTools ツールファミリーを用いたデバッグをより簡素化し、より便利にする目的で、FLASH プログラミング機能を拡張。FLASH メモリへのソフトウェアブレークポイントの挿入、及び FLASH 上でのコードパッチが可能になりました。

オンチップデバッグインタフェースのみを使ってデバッグが可能なプロセッサの殆どは、オンチップブレークポイント数は限定されています。　このため、FLASH メモリ上のソフトウェアをテストしたい開発者は、すぐにこの限界と遭遇することになります。

ローターバッハではこの限界を取り除くため、2002 年より FLASH メモリへのソフトウェアブレークポイント挿入機能を提供しています。また、再プログラムに費やされる時間を最小化するために、TRACE32-PowerView ソフトウェアは現在、全ての FLASH タイプの全セクタリングをサポートしています。

全ての FLASH タイプの全セクタリングサポートにて、FLASH 上のコードにパッチを短時間で挿入することが可能となりました。両機能とも内部及び外部 FLASH メモリに対応可能で、ターゲットベースの FLASH プログラミングにも簡単に使用することができます。

PowerIntegrator

Embedded world 2003(ニュルンベルク/ドイツ、2月)に、ローターバッハは、新しい概念を持つロジックアナライザ、PowerIntegrator を発表しました。 PowerIntegrator はデバッグ中に、ターゲットシステムの全てのバス動作とポートをモニタすることを可能とし、TRACE32 開発環境に簡単に統合化できます。

PowerIntegrator には直ちに立ち上げ、使用可能にするためのサポートパッケージが用意されています。

- **Board Support Packages：**

PowerIntegrator を標準評価ボード用に構成。

- **Bus Support Packages ：**

PowerIntegrator を SDRAM バスや DDR バスのような特定のバスプロトコル用に構成。

PowerIntegratorは自由に設定可能な204チャンネルのデータ入力を有しています。 その内の12チャンネルはクロック入力として使用することも可能です。アクティブプローブでは、次の最高サンプリング周波数を使用できます。

・500MHz／タイミングモード

・200MHz／ステートモード

信号用プローブには次の2種類のいずれかを使用します。

・Mictor プローブ：32 データチャンネルと 2 clock/data チャンネル

・標準プローブ ：16 データチャンネルと 1 clock/data チャンネルを

データサンプリングのスレッショルドは、0～4[V]の範囲内で選択することができます。

PowerIntegrator は、256k フレームのトレースメモリを内蔵、入力データ 48 ビット幅，8ns 分解能のタイムスタンプ付きでトレースメモリにエントリされます。

次の例では、PowerIntegrator の動作を簡単に説明し、SDRAM バスを介したプログラムとデータのトレース例を図に示します。

1．制御信号の構成、及びアドレスバスとデータバスを形成する信号の定義

PowerIntegrator には、個々の信号をグループ化し、ロジック名を割り当てるための汎用"NAME"コマンドがあります。

2．SDRAM バスサイクルのプリフィルタリング

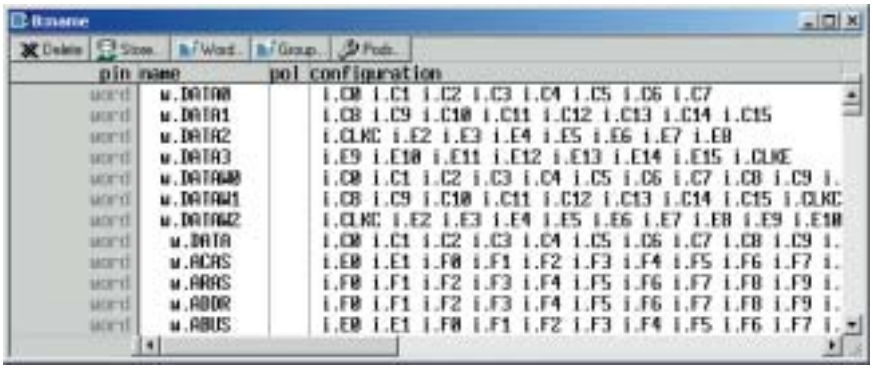
SDRAM バス上の動作からプログラムとデータのフロー生成に使用可能な信号のみを取り出し、トレースメモリに転送することが望まれます。 このプリフィルタリングは SDRAM バスの制御信号を適切な信号レベルに設定することにより可能となります。

3．スクリプトを使用してプログラムとデータフローを生成

次に、プリフィルタにてトレースメモリ上に記録された情報から、プログラムとデータフローを生成します。 このためにスクリプト言語を使用し、フェッチ，リード，ライトサイクルを示す制御信号を定義します。

ページ９のスクリーンショットはこの手順を表しています。

Powerintegrator をデバッガと連動させることにより、フェッチサイクルからのデータ情報を逆アセンブルし、さらには関連する高級言語情報も表示することができます。



トレースデータからプログラムとデータのフローを生成するスクリプトを用意する作業は通常、時間がかかり、かつバスプロトコルに関する知識が必要となります。
ローターバッハはこのソフトウェアベースのソリューションに代わる、バスサポートパッケージの手法を提供します。 SDRAM バスサポートパケージでは上記の 3 つのステップを実践する必要はなく、代わりに、プログラムとデータのフローを生成するためのすべての情報は FPGA にロードされます。FPGA 内のロジックが SDRAM バス上の動作から、アドレス，データ，制御信号を生成します。

サポートパッケージの利点は以下の通りです。

・PowerIntegrator は事前に要求されるトレース仕様に完全に構成されます。

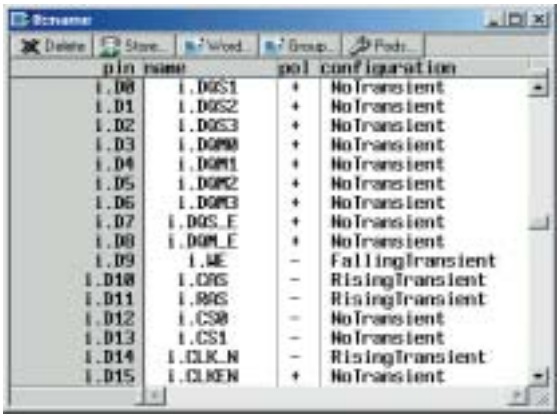
・バスサイクルのフィルタリングをバスプロトコルに対応して最適化します。

・アドレス，データバス及び制御信号を物理的に生成しますので、高度なリアルタイムトリガが可能になります。プログラムとデータのフローから特定のサイクルを選択してトリガしたり、定義されたトリガ条件にてプログラムを停止させることもできます。

PowerIntegrator の開発の目的は、組込み開発市場に汎用トレースソリューションを提供することです。 **ソフトウェアベースのアプローチ**ではスクリプトの使用により、バス動作はもとより全ポートラインの処理が可能となる高度な柔軟性とオープンなソリューションを提供します 。**FPGA ベースのアプローチ**およびサポートパッケージを使用する場合では、バスおよびポート情報の効率的な処理をより速く利用することが可能になります。

ARM、MIPS、及び XScale アーキテクチャの標準評価ボード用ボードサポートパッケージは、本年中に順次リリースされます。 PowerQUICC III 用サポートは最初の評価ボードのリリース後直ちに提供する予定です。

最初のバスサポートパッケージは SDRAM バスになります。 その後にその他のバスアーキテクチャ用パッケージを順次リリースします。 サポートパッケージの対応はマーケットからのフィードバックを基に決定します。



ハードウェアベースドコードカバレッジ

ハードウェアベースのコードカバレッジは、トレースベースのコードカバレッジと比較して、どのような利点をもたらしてくれるのでしょうか。

トレースベースのコードカバレッジは、トレースバッファのサイズによって制限され、短いモニタ時間に対してのみカバレッジ情報の取得ができます。
これに対して、ハードウェアベースのコードカバレッジは、大容量プログラムエリアの長期間のテストにも使用できます。

ハードウェアベースドカバレッジとは？

2002 年 10 月より、PowerTrace にハードウェアベースカバレッジを組み込むことができるようになりました。　このカバレッジ機能の実現のために、PowerTrace には 2 ビット/命令のメモリが付加されています。これら二つのビットは、次の事象をマークするために使われます。

- 命令がプログラムによって実行されたか
- 分岐が実行されたかどうか

	executed	branch taken
ever	0	0
aken	0	1
ot taken	1	0
k	1	1

ハードウェアベースコードカバレッジがサポートされているプロセッサタイプは

ハードウェアベースドコードカバレッジは、現在、ARM-ETM(ARM7/9 ベースのコア)を持つ全てのプロセッサ、また、SH4(日立)、ST40(ST マイクロエレクトロニクス)においても利用可能です。　その他のアーキテクチャへの拡張もまた、常に評価を進めています。

基本的な要件として、実行中にプログラムフローから提供されるトレース情報が、直接および間接分岐しそれぞれの分岐先を含むようにプロセッサを設定することが可能でなければなりません。分岐先情報は通常、コード化され、遅れて提供されるので、各命令上にプログラム実行マーカーを設定するために、FPGA によるトレース情報の付加的な処理が必要になります。

モニタ可能な最大命令数は？

PowerTrace には、4X2M 命令分の記憶スペースがあります。　ハードウェアベースコードカバレッジが実行される最大物理アドレス範囲は、基本的には、各命令の最小幅に依存します。　例えば、32 ビットの命令長であれば、4X8MB バイト分の命令を解析することが可能となります。

USB2.0

2003 年 3 月より、PowerDebug モジュール/イーサネットは、USB2.0 インタフェースを装備して出荷されています。

480Mbps の転送速度を持つ USB2.0 は、100-MBit イーサネットインタフェースと同等の速さでデータ転送します。

PowerProbe の新機能

PowerProbe はプログラムおよびデータフローのトレースを記録するために構成が可能となりました。必要な設定は次の通りです。

1. アドレスとデータバスの信号および制御信号の構成を定義
2. フェッチ、リードおよびライトサイクルに対し、どの制御信号をアクティブにするか定義するスクリプトを用い、プログラムおよびデータフローを生成

THE PRODUCT LINE

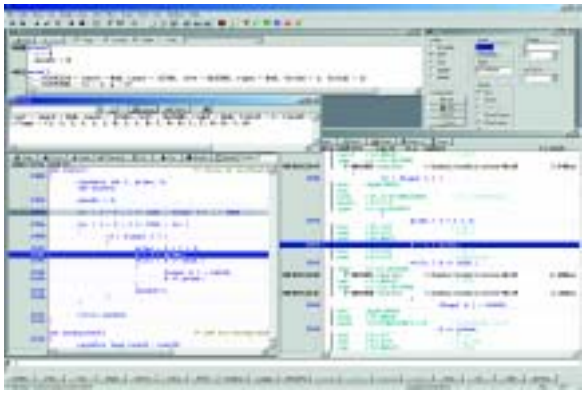
TRACE32-POWERTOOLS

NEXUS for Super10

ST マイクロエレクトロニクスの Super10 用 JTAG デバッガを 2002 年 10 月より提供していますが、この度 NEXUS デバッガ用開発プログラムも完了しました。

JTAG Super10 は全ての標準コンパイラに対応、オンチップブレークとトリガオプションがフルサポートされた使い易いデバッグ環境を提供しています。

NEXUS-Super10 はデバッグ機能に加え、プログラムおよびデータフローのトレース機能も備えています。 NEXUS-Super10 は、トレース解析に加え、実行時間測定、パフォーマンス解析、およびコードカバレッジ測定をサポートする高性能 PowerTrace システムにインプルメントされています。Super10 ファミリーの中の ST10R303 が最初にサポートされるプロセッサです。



TRACE32 の新しいパッケージ

高速化およびトレースシステムの機能強化の要求にお応えするために、ローターバッハは製品開発を継続して行っています。 2003 年 3 月から以下の TRACE32 製品が新しいパッケージデザインになりました。

- デバッグケーブル
- プリプロセッサ
- NEXUS アダプタ

デバッグケーブル

ドングルの形状を大きくし、オンチップデバッグインタフェース制御のために必要となりつつあった、より高度なロジックのためのスペースを確保しました。

新パッケージではデバッグケーブルはドングルにはんだ付けせずに、簡単にプラグインできます。これにより、ケーブル長により柔軟性を持つことができ、ケーブル長も考慮に入れて、オンチップデバッグインタフェースの周波数の選定が可能となります。

プリプロセッサおよびアダプタ

プリプロセッサと NEXUS アダプタを、ターゲットシステムとの間での短絡の防止や汚れ、静電気からの保護のためにケースに組み込みました。
簡単にターゲットシステムに接続することができるフレックス・エクステンションもまた、多くのプリプロセッサと NEXUS アダプタで利用することが可能になりました。

Further information on the internet www.lauterbach.com

TRACE32-FIRE 最新情報

TRACE32-FIRE でサポートされる 16 ビットマイコンの数が 2003 年中に拡大されます。

512K フレームトレースメモリ

2003 年 3 月以降、TRACE32-FIRE は 512K フレームのトレースメモリをご利用になることが可能です。

日立 H8S

TRACE32-FIRE は 2003 年に H8S ファミリーのサポート範囲を拡張します。H8S/2612 と H8S/2678 のサポートにつきましては、2003 年春を予定しています。

日立 H8

2002 年春から、TRACE32-FIRE RISC エミュレータは、日立 H8 ファミリーのサポートを始めました。

下記のプロセッサが現在サポートされています。

・H8/3006/3007/3008

・H8/3044/3045/3046/3047/3048

・H8/3052

・H8/3060/3061/3062/3064/3065/3066/3067/3068

インフィニオン XC16x

2002 年、インフィニオン XC16x ファミリーの新しい派生デバイスのサポートを発表しました。現在、下記のプロセッサ用エミュレータをご提供しています。

・XC161CJ

・XC164CS

2003 年中に、全ての新しいファミリーメンバーをサポートするように、TRACE32-ICD インサーキットデバッガと TRACE32-FIRE RISC エミュレータを対応させるプランを立てています。

モトローラ MCS12

モトローラ 68HC12, MCS12 ファミリーのサポートが 2002 年に完了しました。 68HC912DT128, MCS12DB128 用に新モジュールが追加されました。

ST マイクロエレクトロニクス ST10F276

ST10 ファミリー用 TRACE32-FIRE RISC エミュレータでは、ST10F276 もサポートしています。
ST10F276 は、多くの開発で利用され魅力的なソリューションである ST10F269 よりも巨大なオンチップ FLASH メモリを内蔵しています。 ST10F269 用 TRACE32-FIRE をご使用になられているお客様におきましては、ST10F276 をカバーするために、弊社にて改造を承ります。

LAUTERBACH