

論文

バススヌーピング方式によるリアルタイムタスクトレーサ IP の開発

武田 有志*¹⁾ 岡部 忠*¹⁾ 仲村 将司*¹⁾ 佐藤 研*²⁾

Development of a real-time task tracer IP based on bus snooping methods

Yuji Takeda*¹⁾, Tadashi Okabe*¹⁾, Masashi Nakamura*¹⁾, Ken Sato*²⁾

Recently, the use of RTOS is advanced in a multi-core processor on FPGA/SoC, and to watch the task transitions is important when confirming dead-lock conditions and operations of real-time processing. However, the conventional methods which use software hook processes or processor specific hardware, i.e. the ICE, are not suitable for lengthy real-time monitoring without processor loading. Then, we have developed a task tracer IP to extract only task transitions by snooping a general-purpose bus of FPGA/SoC. This IP consists of several taps and a master, and the selectivity of communication I/F that transmits the trace data and the core scalability are improved. Moreover, it can compress the trace data through difference operations and variable length coding. A 46.3% reduction in traffic was confirmed as a result of actually tracing the TCB access by the TOPPERS kernel.

キーワード：FPGA/SoC, マルチコアプロセッサ, リアルタイム OS, タスクトレース, データ圧縮

Keywords：FPGA/SoC, multi-core processor, RTOS (Real-Time OS), task trace, data compression

1. まえがき

近年の組込みシステムでは、イーサネットや USB 等の様々な通信処理が必須であり、RTOS (Real-Time OS) を使ったアプリケーション開発が進められている。一方、FPGA/SoC では、任意数のプロセッサコアを搭載できるため、既に構築されたシステムへの機能の拡張として、マルチコア化するための開発基盤が整えられつつある。RTOS の挙動を把握する方法としては、タスクトレース⁽¹⁾が有用であり、デッドロックやリアルタイム性の確認に不可欠である。また、トレース結果は、プロセッサ負荷の空き状況を解析することで、コアへのタスクの振り分け方や、実装チップを低コスト化するための指標に活用できる。

本研究では、FPGA/SoC の汎用バスを監視して、タスク遷移を抽出してリアルタイムに監視できるトレーサ IP を開発した。本 IP はタップとマスタで構成され、タップとマスタ間には FPGA/SoC の汎用バスで接続される。そのため、タップを追加することで任意のコア数に対応できる。また、トレース情報を送信する通信 I/F 用コントローラについてもマスタと汎用バスで接続されており、通信量の増大に柔軟に対応できる。さらに、本マスタでは、トレース情報の差分化と可変長化によるデータ圧縮を行っており、通信量を削減することが可能である。

2. 従来のトレース方法

従来のトレース方法は、ソフトウェアによるものと、専

用ハードウェアによるものの 2 つに分類できる。ソフトウェアによる方法では、OS のサービスクール⁽²⁾の呼出し前後に、トレース情報を書き出すための処理を挿入することで実現しており、トレース条件等を自由に設定できるという利点がある。しかし、この方法では、プロセッサに負荷が伴うことから、トレース処理の有無によって異なる動作を引き起こす。また、プロセッサ自体が何らかの要因によってトレース処理を実行できない状態に陥った場合には、ソフトウェア的な要因であるのかハードウェア的な要因であるのか、原因究明が困難である。

一方、専用ハードウェアによる方法では、プロセッサそのものを、トレース機能を持ったプロセッサに置き換えることで実現している。一般的には ICE (In Circuit Emulator) と呼ばれる。ソフトウェアによる方式と比べて前述の問題は発生しないが、プロセッサで実行されるすべての命令がトレースされることから、粒度が細かい反面、プロセッサの処理速度に追従できる高速なインタフェースとメモリが必要となるため、一般的には高コストである。また、メモリ量に制限があることから、製品出荷前などの連続的なテストには不向きである。

FPGA/SoC では、トレース機能に JTAG を利用した実現例が存在する⁽²⁾。しかし、JTAG は低速であり、FPGA の内部メモリに一度蓄積する方式が採られている。よって、前述の専用ハードウェアによるものと機本的には同じ原理である。

3. バススヌーピング方式によるトレーサ IP

我々が開発したトレーサ IP の設計指針は、次に示す通り

*¹⁾ 情報技術グループ

*²⁾ 電子・機械グループ

である。

(1) コアへの負荷をゼロ ソフトウェアによるトレースでは、トレース情報を一度内部のリングバッファ (FIFO) に蓄積し、通信処理が行われる。この部分をハードウェア化、コアとは別に、並列動作させることで実現する。また、トレース情報の取得には、バスアクセスと同期して取り出すように実現する。

(2) スケーラビリティ確保 FPGA/SoC では、プロセッサを任意のコア数で構成できることが利点である。その利点を活かすには、コア数増加に対して柔軟に対応することが必要である。よって、FPGA/SoC の汎用バスを活用したアーキテクチャとする。

(3) 通信 I/F の選択性向上 選択性を向上することで、ローカル PC での監視だけでなくリモートでの利用が可能になる。そのためには、通信用コンポーネントとの接続性が重要であり、(2)と同様に汎用バスで構成する。通信の単位データ長は、RS-232C 等のレガシな通信 I/F で利用可能なように1バイト (8ビット) を基準とする。

(4) トレースデータ量増大への対応 マルチコア構成では、シングルコアに比べて単純に N 倍のトレース情報が出力される。これに対応するには、後解析で再現可能なレベルにまでにデータ自体を削減し、また、リアルタイム圧縮を試みる。

図1は、以上の設計指針の元で構築したトレーサ IP のアーキテクチャを示している。トレーサ IP は、マスタとタップ、そして、通信 I/F 用コントローラで構成されている。タップは各コアに接続されており、バスアクセスからトレース情報を取得する機能を持つ。また、マスタは複数のタップを走査し、タップで取得したトレース情報を通信 I/F 用コントローラに受け渡す機能を持つ。

タップとマスタ間、および、マスタと通信 I/F 用コントローラ間は、汎用バスを通じて接続されており、コア数に対するスケーラビリティと通信 I/F の選択性が高められている。すなわち、コア数が増加した場合にはタップを追加することで対応し、トレース情報の転送量が増大した場合にはマスタと通信 I/F を追加することで対応可能となる。今回

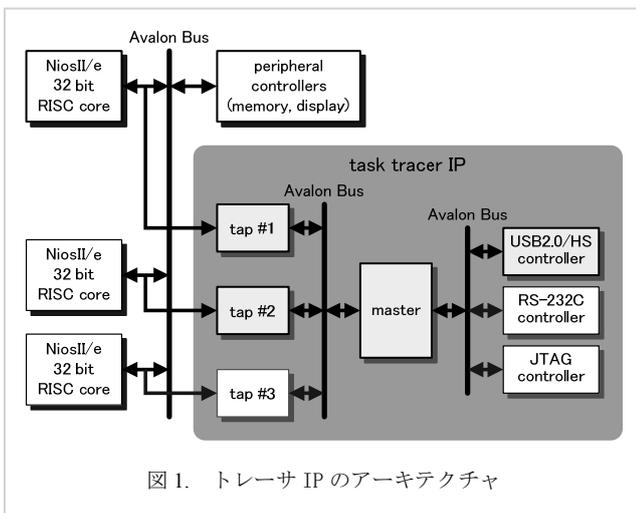


図1. トレーサ IP のアーキテクチャ

の開発では、汎用バスに大手 FPGA ベンダの1つである Altera 社の Avalon Bus⁽³⁾を選定しているが、Xilinx 社の FPGA/SoC においても PLB (Processor Local Bus)⁽⁴⁾を参照することで、同様の構成で実現可能である。

4. 内部構成

4.1 タップ部 図2に示すように、タップ部は命令検出器とデータ検出器、そして、これらの検出結果を結合するバインダと前回値との差分を取るブロックで構成されている。命令検出器およびデータ検出器は、検出アドレス設定レジスタが示すメモリ領域がアクセスされた際に検出信号を出力する。バインダは、検出信号の発行順序によって、命令検出器およびデータ検出器で検出した際のアドレスとデータを保存する。また、命令検出器において検出された時刻は、マスタから供給されるタイムスタンプカウンタから取得され、バインダで得られた情報とともにトレース情報として記録される。トレース情報が記録されるとタップは割り込み信号を発行し、マスタから読み込み要求があるまで現在のトレース情報を保持する。

図3はトレース情報の構造を表しており、32ビットのアドレス/データバスを用いた場合を示している。valid はタップから割り込み信号が発行された後、次のトレース情報で更新される前にマスタによって読み取られたかどうかを表し、情報の受け渡しが間に合っているかどうかの判断に利用できる。tap ID は、トレース情報を取得したタップを識別するための値である。1トレースのデータ長は128ビットであるが、これは Avalon Bus において設定できるバス幅は最大1024ビットであり、1サイクルで十分に読み出せる。

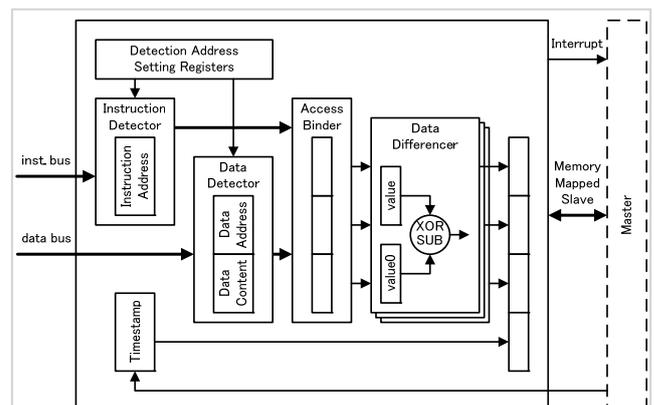


図2. タップ部のハードウェア構成

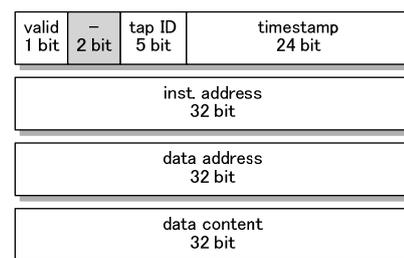


図3. トレース情報のデータ構成

アドレスおよびデータに記録される値は、前回送り出した値との差分である。命令アドレスは、命令キャッシュと同様にアクセスの局所性がある可能性が高く、また、データアドレスは、同一タスク内であれば同じTCBを参照する可能性が高い。したがって、差分としたのは、少ないビット数で表せる可能性が高いからである。一方、タイムスタンプでも同様のことが言えるが、タイムスタンプは、複数のタップから集められることに着目すると、マスタで差分を取った方が小さくなるため、差分処理はマスタで実施する。差分の演算方法としては、代表的な排他的論理和もしくは減算によって行い⁽⁵⁾、実トレースから最適な方法を選択する。

4.2 マスタ部 図4に示すように、マスタ部はタップからトレース情報を一時蓄積するためのFIFO、タイムスタンプに対して差分を取るためのブロック、トレース情報を8ビット単位の可変長データに変換するブロック、そして、全タップに現在時刻を供給するタイムスタンプカウンタから構成されている。一方、図5は、可変長データに変換する際の流れを示している。送信時において7ビット単位で区切られた各データの先頭には、下位の値が存在するか否かを示す1ビット値が付加され、可変長データの終端が判別できる仕組みとなっている。

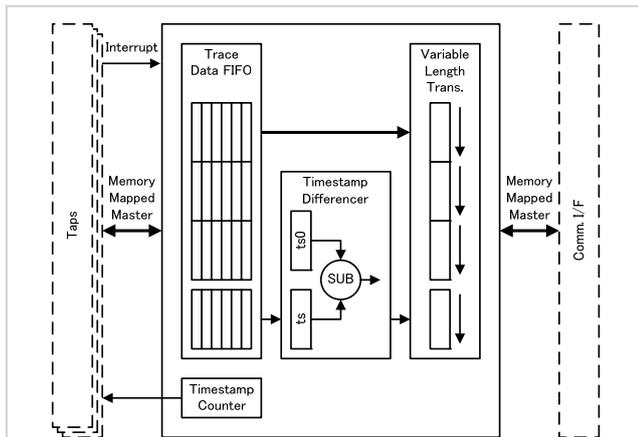


図4. タップ部のハードウェア構成

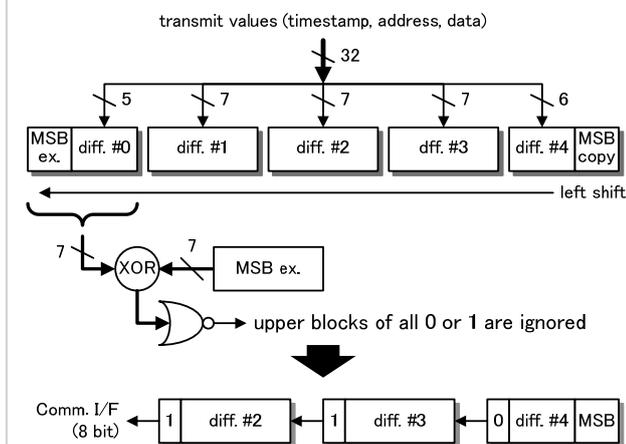


図5. トレース情報の圧縮

差分で表されたすべてのデータは、上位側のビット列に1または0が連続して現れることが期待される。そこで、図5の圧縮ではデータを上位から7ビット区切りで走査し、同じ値のビットが存在する限り当該バイトを送信しない流れとなっている。最後に送信する1バイトには、送信されなかった上位ビットを復元するために、MSBの1ビットが付加される。

5. 評価

開発したトレーサIPを用い、実際にTCB全領域を対象としたタスクトレースを行った。タスク遷移のみを取得する場合にはTCB全領域である必要性は無いが、オフセット演算が必要となりハードウェア量が増えるため、そのように設定している。

5.1 実装したFPGA環境 実装したFPGAボードは、terasic社DE1であり、Cyclone II EP2C20F484が搭載されている。プロセッサは、Altera社のNios II/e×1、メインメモリであるSDRAM、そして、いくつかのペリフェラルで構成され、Nios II/eとSDRAMのクロック周波数は、それぞれ100MHz、50MHzである。また、トレース情報を取り出す通信I/Fには、FTDI社のFT2232Hを用いた⁽⁶⁾。FT2232Hは、パソコンでの仮想シリアル通信が可能なUSB2.0 High Speedの汎用チップであり、最大の実効転送レートは220Mbpsである。

使用したRTOSは、μITRONとして有名なTOPPERS/JSPおよびTOPPERS/FMP⁽⁷⁾である。アプリケーションプログラムは、温度センサから得られた温度データをRS-232Cで送信するものであり、タスク構成は、LED、LCD等の表示、RS-232C等の通信、そして、A/D変換、の複数から成る。タスクの実行タイミングとしては、カーネルのディスパッチ単位時間1msec内に複数タスクが同時にディスパッチできる状況を設定した。

5.2 対応可能コア数 最大のデータ転送量を計測するため、タスク遷移間の最大トレース数を求めた。その結果、87.3トレース/msecであることが確認された。1トレ

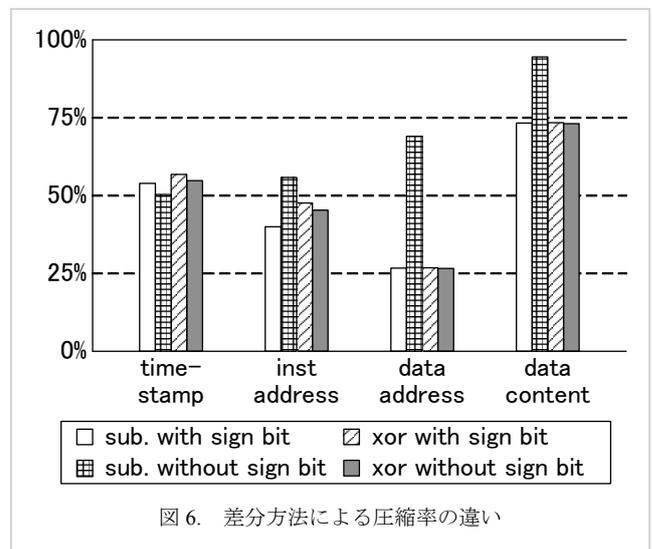


図6. 差分方法による圧縮率の違い

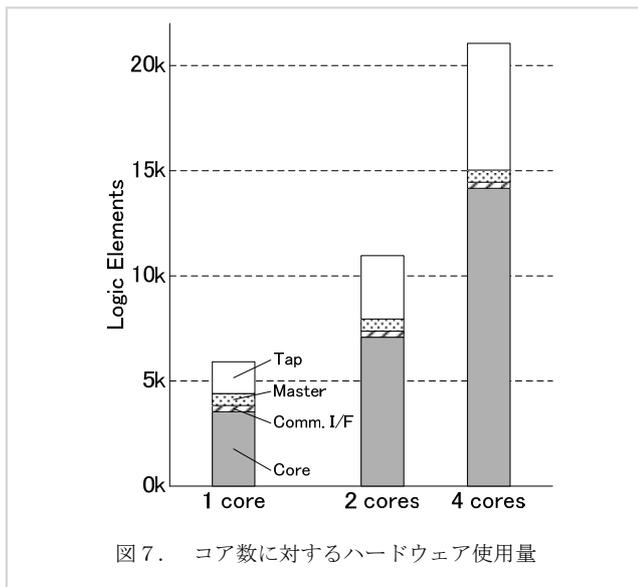


図7. コア数に対するハードウェア使用量

は、図3に示すように16バイトで構成されていることから、1コア当たりの通信速度は11.2Mbpsとなる。FT2232Hの最大転送速度は220Mbpsであることから、データ長が128ビット固定の場合には、19.6コアまで対応可能である。

一方、図6は、トレース情報の各値に対する差分化の方法として、減算(sub)、排他的論理和(xor)および符号ビット有無を選択することにより圧縮率に違いが現れるかを示している。この結果、タイムスタンプとアドレスに対してデータ圧縮が有効であることが分かる。

圧縮率が最大となる組合せでは46.3%の削減、ハードウェア量の抑制に全て符号ビット有ったとした圧縮率が最大となる組合せでは45.2%の削減が可能である。したがって、今回のFPGA環境においては、35.8コアまで対応できる。以上の結果から、高コストのICEを利用せずに、安価な汎用USB2.0チップでのマルチコアのタスクトレースが可能である。

5.3 ハードウェア量 図7は、コア数に対するハードウェア使用量を示している。論理合成には、Quartus II 10.0 Build 262を用いた。タップは、コアに対して46.6%のハードウェア量が必要であり、マスタと比べて3倍程度となっている。今回使用したEP2C20F484は、20k個のLogic Elementsを有しており、最大3コアまで構成することが可能である。EP2C20F484は比較的小規模のFPGAである。以上の結果から、開発したトレーサIPは、比較的小規模のFPGAにおいても十分に1チップ内に収めて実装できることが確認された。

6. おわりに

本論文では、次の特徴を有するRTOS向けトレーサIPを開発した結果について述べた。

- コアに対する負荷をゼロとするためバススヌーピング方式を採用。
- コア数増加に対するスケーラビリティを確保し、通信I/Fの選択性を向上するためのFPGA/SoCの汎用バス

を活用したアーキテクチャ。

- トレースデータ量の増大に対応するためのトレース情報の圧縮。

さらなる通信量の削減としては、次の方法が考えられる。今回の実装ではTCB領域全体に対してトレースを行ったが、タスクがディスパッチされるのは、タスク制御ブロック(TCB)に対してスタックポインタとプログラムカウンタが(TCB_sp)(TCB_pc)からロード命令によって復元される時である。今回の実装におけるTCB領域の場合、JSPでは32バイトごとに、FMPでは80バイトごとに連続して並んでいる。そこで、さらに通信量を削減するには、TCB内のオフセットを計算し、そのオフセットが前述のロード命令のオフセットであるか否かを判断できるテーブルを用意することで実現できる。

また、ディスパッチされているタスクでは、TCBのアクセス順序に偏りがある。よって、生起確率が最も高い次のアクセスをテーブルで表現し、予測から外れたときだけデータを送ることで、データアドレスの転送を省略できる。また、データ値は、リード/ライトの両方がトレースされる仕組みとなっているため冗長性が高い。よって、データ値に対するキャッシュを用意し、同じ値が来たかどうかを判定して、データ転送を省くことができる。

今後の発展性としては、次のことが挙げられる。本IPではトレース情報を汎用バスで受け渡すことが可能である。よって、トレース情報をFPGA内部で参照することにより、タスクが正常なサイクルで動作しているか、自己監視に利用できる。また、近年のセキュリティ問題により、OSに対する更新が頻繁に起こっている今日、更新後のOSがタイミング的に許容範囲内に収まるかを検証する際に有効である。

(平成23年5月19日受付、平成23年7月1日再受付)

文 献

- (1) 後藤 隼式, 本田晋也, 長尾卓哉, 高田広章: “トレースログ可視化ツールの開発”, 情報処理学会組込みシステム研究会研究報告, Vol. 2009-EMB-012, No. 22, pp. 73-78 (2009)
- (2) 入月康晴, 大原衛, 坂巻佳壽美: “JTAGを用いた組込みシステムのオンライン自己監視手法”, 日本信頼性学会誌「信頼性」, Vol. 32, No. 3, pp. 185-190 (2010)
- (3) Altera: “Avalon Interface Specification” (May 2011), http://www.altera.com/literature/manual/mnl_avalon_spec.pdf (2011/05/19)
- (4) Xilinx: “LogiCORE IP Processor Local Bus v4.6” (Sep. 2010), http://www.xilinx.com/support/documentation/ip_documentation/plb_v46.pdf (2011/05/19)
- (5) Kentaro Sano, Kazuya Katahira and Satoru Yamamoto, “Segment-Parallel Predictor for FPGA-based Hardware Compressor and Decompressor of Floating-Point Data Streams to Enhance Memory I/O Bandwidth”, IEEE Data Compression Conference, pp. 416-425 (2010)
- (6) Future Technology Devices International: “FT2232H”, <http://www.ftdichip.com/Products/ICs/FT2232H.htm> (2011/05/19)
- (7) TOPPERS プロジェクト: “TOPPERS プロジェクト/FMP カーネル”, <http://www.toppers.jp/fmp-kernel.html> (2011/05/19)