FPGA を用いた D3 通信制御処理ソフトウェアのハードウェア化と高信頼化

A Hardware Implementation of D3 Communication Control using FPGA for its Dependability

○正 中坊 嘉宏, 北 勝也 (産総研), 齋藤 元 (GRX)

Yoshihiro NAKABO, AIST, nakabo-yoshihiro@aist.go.jp
Katsuya KITA, AIST
Hajime SAITO, General Robotics Corporation

There is increasing demand for personal care robots which require a high level of safety and dependability. We have developed a D3 module with a Time Triggered Protocol (TTP) which realizes synchronous and distributed control systems for such dependable robots. However, a CPU in an FPGA chip used for a part of a safety related system of an existing D3 module is difficult to be complied with IEC61508 functional safety because of its complexity. To solve this problem, instead of using CPUs for the safety related systems, we have introduced a hardware circuit directly derived from D3 driver software by using technique of high-level synthesis, which makes easier to prove the equivalence of its implementation and the safety requirements.

Key Words: Personal care robots, Functional safety, Time Triggered Protocol, Distributed control, High-level synthesis

1. はじめに

近年, 日常空間で利用する生活支援ロボットや人と共存して生産を行う次世代産業用ロボットの実用化が期待されている。しかし, これらのロボットは人と近接して動作するため, 安全の問題を解決する必要がある。我々は, このような高信頼, かつ安全なロボットのための同期分散制御システムとして Time Triggered Protocol (TTP) 通信の IP コアを Field Programmable Gate Array (FPGA) に内蔵した D3 モジュールを開発している。しかし, これまではモジュールの安全関連系の一部に CPU が組み込まれていたため, 機能安全認証が困難であった。そこで本研究ではソフトウェアから, 高位合成の技術を用いて, 直接 FPGA のハードウェア回路を生成し, 安全要求仕様との等価性を検証することで機能安全への適合を目指した。

2. TTP 通信の実装方法と機能安全への適合

2.1 高信頼同期通信 D3 モジュール

我々はこれまでに, 高信頼な同期分散制御系として D3 プラットフォームを開発し TTP 通信を用いたロボット制御を実現している[1]。D3 プラットフォームは, 決定論的 (Deterministic) 動作を可能にした高信頼 (Dependable) の分散 (Distributed) 制御システム用のプラットフォームで, その構成要素である D3 モジュールは, 分散制御系の分散ノードとして FPGA を搭載し, センサやモータの制御, 入出力を行うことが可能である。Fig. 1 の D3 モジュールの写真では, 4cm 角サイズの 4 つのモジュールが示されている。

通信制御は, FPGA に内蔵した TTP/C 通信コアにより, IEC61508 機能安全から DO-178b, DO-254 などの航空安全規格まで, 認証に対応した通信プロトコルが利用できる。通信の物理層は RS485 の冗長 2ch で 5Mbps, もしくは非冗長で 2 倍の 10Mbps 通信容量が可能である。

2.2 従来の一般的な TTP 通信の実装方法

ここで, 前節で説明した D3 モジュールでの TTP 通信の実装と, 従来の一般的な TTP 通信の実装方法, そして

Fig. 1 TTP 通信を用いた D3 モジュール

本研究で提案するハードウェア回路による通信制御の実装について, 比較して説明する。

まず従来の TTTech 社による一般的な TTP 通信の実装では[2], Fig. 2 に示すように, TTP の通信プロトコルを処理する C2NF コアは AS8202NF という独立した IC チップとして実装され, ユーザアプリケーションから通信にアクセスするためのドライバ(主に FT-COM と呼ばれる部分)は, TTP-OS でのみ動作することになっていた。この結果, ユーザのアプリケーションも同様に TTP-OS でのみ動作することになり, また対応する CPU が, 現在は MPC555 などに限られているため, これ以外の CPU を使うには新たに TTP-OS の移植から行う必要があった。安全関連系には CPU と OS の両方が含まれ, そのため TTP-OS は機能安全認証可能な OS となっている。

2.3 これまでの D3 モジュールでの TTP 通信の実装方法

次にこれまでの D3 モジュールの TTP 通信の実装方法を説明する。これまでの D3 モジュールでは, Fig. 3 に示すように, TTP 通信を処理する C2NF コアは IP コアとして FPGA に内蔵され, また通信ドライバとユーザアプリケーションは FPGA のソフトコア CPU, 具体的には Xilinx 社 FPGA の MicroBlaze 上で動作していた。また安全関連系として通信部分のみで完結するための安全ロジックも同時に組み込むこととしていた。

ただしこの実装方法では, FPGA に加えてソフトコア

Fig. 2 一般の TTP 通信の構成

Fig. 3 これまでの D3 通信モジュールの構成

Fig. 4 提案する通信ドライバのハードウェア化

CPUの機能安全認証が必要となる。実際には Xilinx 社を含めて各社ソフトコア CPU はそのままでは機能安全に対応しておらず、ユーザ自身で自己診断機能を組み込むなどする必要があった。

2.4 TTP 通信のハードウェア実装の提案

最後に、本研究で提案する、ハードウェアによる TTP 通信処理の実装を説明する。本研究では、これまでの D3 モジュールの通信ドライバと安全ロジックの部分で、できる限りそのままハードウェア回路 (順序回路) として FPGA に実装する。その際、従来はソフトウェアとして作成されたものを、それと等価なハードウェアとするために、C プログラムを人力とする高位合成 [3]と、アサーションによる適合性検証および形式手法による等価性検証 [4]を行う。これを Fig. 4 に示す。これによりソフトコア CPU は安全関連系から除外され、認証の必要がなくなる。

今回は、3 章で述べるように、動作試験による検証とアサーションの設計までを行った。次節では、本研究で提案するハードウェア回路による実装の利点を説明する。

2.5 機能安全における CPU とハードウェア回路の比較

例えば文献 [5]には、DO-178b, DO-254 などの航空安全規格において、一般に Complex Electronic Hardware とされる CPU システムと、Simple Electronic Hardware とされる順序回路とで、安全性の検証方法が大きく異なることが示されている。この区別は IEC61508 機能安全でも同様である。

具体的には、Fig. 5 に示すように、CPU システムではメモリに格納されたプログラムやデータによって ALU での演算とバスを通したデータの受け渡しが行われ、再びメモリに格納される。このように処理が再帰的に行われ、そのたび

Fig. 5 CPU によるシステム構成

Fig. 6 順序回路によるシステム構成

に結果が分岐するため、システムとしては、回路規模と比較して無限とも言える状態をとることになる。すなわち、機能安全における安全状態を確認するうえで、とりうる全ての状態を検証することが困難である。そのため、通常 CPU の安全性を検証するには、内部を部分ごとに分けた上で、部分の誤り (エラー) が全体に影響を与えないかどうか、エラーを前提とした診断可能性による検証にならざるを得ない。

一方、Fig. 6 に示す順序回路によるシステムでは、組み合わせ回路による処理結果がレジスタに戻るものの、処理の内容があらかじめ決まっているために、その状態数はレジスタの数に応じた限られたものになる。そのため、例えば規模にもよるが、形式手法を使って全ての状態を網羅し、安全性を完全に証明することも可能である。

2.6 FPGA の機能安全

最後に FPGA の機能安全について述べる。Fig. 4 でも分かるように、提案する方法では、ハードウェアとしての FPGA の機能安全対応が必要となる。これについては、文献 [5,6,7] に示すように、回路の冗長化や誤り訂正、また定期的な FPGA の再構成などの手法が示されている。

3. 高位合成による開発と検証

3.1 高位合成による開発フロー

今回 D3 モジュール 4 つからなるサンプルシステムのアプリケーションをターゲットとして、高位合成を行った。具体的な開発フローを Fig. 6 に示す。

黄色のボックスが、各開発ツールを用いた作業となる。図の中央のフローが通信ドライバの高位合成の部分で、認証済みのツールである TTTEch 社のツールから生成されるドライバに対して、3.3 節で説明する修正を行った上で Mentor 社のツールにより高位合成を行った。また左側のフローは、例えば 3.2 節で説明するような、CPU を用いない

Fig. 7 高位合成の開発フロー

ことによる調停のための回路設計の部分である。右側のフローは、従来と同様のユーザアプリケーション用の CPU 開発フローであり、これら 3 つの部分で Xilinx 社のツールにて統合し、最終的な FPGA 回路イメージを生成した。

3.2 タイミング設計

TTP の通信の IP コアである C2NF へのアクセスは共有メモリとなっており、CPU を通じたアクセスを前提としている。一方、ソフトウェアから高位合成した回路は、メモリなどの存在を前提としておらず、処理のタイミングですぐにデータにアクセスしようとする。そこでこのギャップを埋めるために、CPU のメモリアccessを模擬してタイミングを明示的に設計し、回路として実装する必要があった。

例として ROM の動作シーケンスを Fig. 8 に示す。実際の ROM の動作シーケンスをあらかじめ定義し、ソフトウェアから合成した回路が ROM のデータにアクセスする際には、このシーケンスを考慮したウェイトの回路を追加して入れておく必要がある。

3.3 FPGA リソースの削減

高位合成では、ソフトウェアの処理ひとつひとつがすべてそのままハードウェア回路に置き換わり、CPU 回路のようなリソースの再利用が行われない。そのため、回路規模が簡単に巨大化してしまう。

本研究においても、当初、TTP ツールで生成したソースコードをそのまま高位合成してハードウェア化し、シミュレーションで正常動作を確認したが、いざ FPGA に実装しようとなると回路規模が大きすぎて実装不可能であった。そこで、高位合成で生成される回路規模を抑えるように、ソースコードで、冗長な処理を圧縮する修正を行った。この結果、既存の D3 モジュールに搭載済みの FPGA チップへの実装が可能となった。今回はこの修正を手動で行っ

Fig. 8 ROM の動作シーケンス

たが、今後は、正しさの検証のためにもルールベースによる自動化を行うことが有効だと考えられる。

3.4 実装と検証結果

以上の開発を行い、実際に D3 モジュールに実装して動作検証を行った。ユーザアプリケーションレベルの検証として、プログラムによるデータの送受信と同期制御を行い、これまでの D3 モジュール 4 つのサンプルシステムと同様な結果を得ることができた。すなわち従来の D3 モジュールと等価なハードウェアシステムが実現できていることが確認できた。

3.5 アサートの設計

前節では動作テストによる検証を行ったが、厳密な等価性と妥当性を検証するには、アサートによる検証や形式手法による検証が必要である。今回、アサートの設計までを行った。例として ROM のアサートの設計結果を Table 1 に示す。

4. おわりに

D3 モジュールにおける CPU の機能安全対応の問題を解決するため、高位合成を用いたソフトウェアのハードウェア化を提案した。また実際に開発フローに基づいて FPGA への実装を行い、動作テストによりソフトウェアと等価なハードウェアが実現できていることを確認した。

今後の課題として等価性と妥当性の検証が挙げられる。また現在の開発フローに対して、より汎用かつ効率

Table 1 ROM のアサート

アサーション名	説明
CLK	シミュレーション開始から不定ではない
RESETn	シミュレーション開始から不定ではない シミュレーション開始から3サイクル以上アサートされる 一旦ネゲートされると、シミュレーションの終了までネゲートされ続ける
VALID	リセットがアサートされている間はネゲートされる アサートされた時、次のサイクルから READY がアサートされるまでアサートが保持される
ADDR	リセットがネゲートされている間は不定ではない 前のサイクルで VALID がアサートされていて、かつ同じサイクルで READY がネゲートされている時、前の値を保持する
READY	リセットがアサートされている間はネゲートされる アサートされた時、前のサイクルで VALID がアサートされている
RDATA	READY がアサートされている時は、不定ではない

的な開発を行うため、ターゲットとなるシステムごとに関 が必要 部分 となるべく減らし、再 利用 可能な部分を増やすことが有効である。また手動での作業をなくし、自動でフローが進むようにスクリプトなどを作成することも必要である。今後開発を進めたい。

文 献

- [1] 中坊嘉宏, 朴雲埴, 齋藤元, "サービスロボットの安全性向上を目指す D3 プラットフォーム", ロボティクス・メカトロニクス講演会(Robomec2010), 2010 年 6 月, 旭川.
- [2] <http://www.tttech.com/technologies/ttp>
- [3] Synopsis, "Symphony HLS(High-Level Synthesis)ソリューション"
- [4] 中坊嘉宏, 山田陽滋, "人と共存するロボットのための安全関連システムのモデリングと形式手法による検証", 日本ロボット学会誌, 27-8, pp.851-856, 2009/10
- [5] Federal Aviation Administration (FAA) Certification Authorities Software Team (CAST), "Simple Electronic Hardware and RTCA Document DO-254 and EUROCAE Document ED-80, Design Assurance Guidance for Airborne Electronic Hardware", Position Paper CAST-30, August 2007.
- [6] Federal Aviation Administration (FAA) Certification Authorities Software Team (CAST), "Clarifications on the use of RTCA Document DO-254 and EUROCAE Document ED-80, Design Assurance Guidance for Airborne Electronic Hardware", Position Paper CAST-27, June 2006.
- [7] Xilinx Joel Le Mauff and Jeff Elliott, "Meeting DO-254 and ED-80 Guidelines When Using Xilinx FPGAs", Xilinx White Paper, WP332(v.1.0), 2009
- [8] J. Ryan Kenny and Karl-Heinz Gatterer, "DO-254 Support for FPGA Design Flows", Altera White Paper, WP-01065-1.0, 2008