社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2006－SLDM－124 (12)
2006／3／16

プロトコル表現効率化に基づくプロトコル変換器合成手法に関する研究

石川 悠司† 渡邊 翔太†† 瀬戸 謙修††† 藤田 昌宏†††

† 東京大学工学部電子情報工学科
†† 東京大学大学院工学系研究科電子工学専攻
††† 東京大学大規模集積システム設計教育研究センター
E-mail: †yuji@cad.t.u-tokyo.ac.jp, ††shota@cad.t.u-tokyo.ac.jp, †††seto@cad.t.u-tokyo.ac.jp,
††††fujita@ee.t.u-tokyo.ac.jp

あらまし 大規模 LSI 設計においては既存の設計資産（IP）の再利用が重要となってきているが，その際にしばしばインターフェースプロトコルに互換性がとれない事が問題となる．この問題を解決するためには，プロトコル変換器の自動合成手法が必要であるが，多くの既存の手法では対象とするプロトコルの仕様を形式的に表現する工程で人手を消費する．この報告では，プロトコルの仕様を少数の宣言文で簡潔に表現できる表現方法の提案と，提案表現法による仕様記述を入力とするプロトコル変換器の自動合成手法の提案を行う．また，提案手法に基づいてプロトコル変換器を生成し，作業量の削減量を評価した結果を示す．

キーワード IP（設計資産），インターフェースプロトコル，プロトコル変換，プロトコル仕様記述，ラッパ

Protocol Wrapper Generation from Statement Based Specification

Yuji ISHIKAWA†, Shota WATANABE††, Kenshu SETO†††, and Masahiro FUJITA†††

† Department of Information and Communication Engineering, Faculty of Engineering, University of Tokyo
Hongo 7–3–1, Bunkyo-ku, Tokyo, 113-8656 Japan
†† Department of Electronics Engineering, School of Engineering, University of Tokyo
Hongo 7–3–1, Bunkyo-ku, Tokyo, 113-8656 Japan
††† VLSI Design & Education Center, University of Tokyo
Yayoi 2–11–16, Bunkyo-ku, Tokyo, 113–8656 Japan
E-mail: †yuji@cad.t.u-tokyo.ac.jp, ††shota@cad.t.u-tokyo.ac.jp, †††seto@cad.t.u-tokyo.ac.jp,
††††fujita@ee.t.u-tokyo.ac.jp

Abstract IP Reuse is becoming important in VLSI design. When IPs are used in the design, it is often become problem that interface protocol does not match among IPs. To avoid this problem, methods for protocol wrapper synthesis are necessary. However, most of the existing methods require protocol specification in form which consumes lots of time to describe. In this paper, we propose a language to describe protocol specification concisely with a few statements and a protocol wrapper synthesis method which accepts the proposed description language as its input. We demonstrate and evaluate our method by sythesizing some wrappers.

Key words IP(Intellectual Property), Interface Protocol, Protocol Conversion, Protocol Specification, Protocol Wrapper

1. はじめに

現在，半導体の集積・製造技術の進歩は著しく，大規模 LSI 設計の現場では，製造可能なトランジスタ数に設計可能なトランジスタ数が追いつけなくなる事態が生じている．そのため，設計期間の長期化が問題となっている．より短期間の内に，より大規模な回路を設計するため，コンピューター設計支援（CAD）技術の開発と導入が積極的に進められている．

大規模 LSI の設計生産性を上げる方法として，既存の設計資産（IP:Intellectual Property）を別の設計に再利用するという方法がある．この方法では，複数の IP とその間を接続する通信路という構成で要求仕様を満たす設計を実現する．この際に，設計に組み込みたい IP のインターフェースプロトコルが通信路，あるいは他の IP のインターフェースプロトコルと適合しな

い場合があることが問題となる．インターフェースプロトコルの互換性のないIPを接続する際には，プロトコルを変換する論理回路を生成してIPの間に挿入する必要がある．本研究では，あるプロトコルの信号の論理値やタイミングを変換して別のプロトコルにする論理回路をラッパと呼ぶことにする．ラッパの生成は，LSI設計作業において本質的な作業ではないため，ラッパの生成に人手が割かれることは設計生産性が下がることにつながる．このため，ラッパの自動生成技術が必要となる．

2. 関連研究

既存のラッパ生成手法の研究をラッパの構造で分類すると，内部共通プロトコルを経由してプロトコルを変換する手法と2つのプロトコルの仕様からラッパを直接生成する手法の大きく2種類に分類される．前者は，内部共通プロトコルとIP固有プロトコルの間を翻訳するラッパをラッパライブラリに持っておき，ライブラリの要素を組み合わせることでラッパを生成するという手法である．これらの手法を以降，ライブラリベース手法と呼ぶことにする．ライブラリベース手法では，新しいプロトコルを導入する際に内部共通プロトコルとの変換ラッパを作るだけで，ラッパライブラリに登録されている全てのプロトコルとの間のラッパを生成することができる．しかし，ライブラリベース手法の研究では，ライブラリの要素が豊富にあることを前提としており，ライブラリの要素を新規設計する方法は詳しく扱っていないものが多い．また，直接生成する手法の例としては，プロトコル表現をオートマトンの形で受け取り，2つのオートマトンにグラフ探索アルゴリズムを適用することでラッパを生成するという手法がある[1]．

ラッパの自動生成手法を使用する際には，プロトコル変換の対象とするプロトコルの仕様をどのようにして生成手法に与えるかが問題となる．多くの研究ではプロトコルを表現するFSM (Finite State Machine:有限状態機械) またはそれに準ずるものを生成手法の入力としている．FSM以外を入力とする例として，インターフェース部分の記述を含むVerilogの設計記述を入力して，設計記述のインターフェースプロトコルを内部共通プロトコルに変換する手法がある[2]が，正しいラッパを生成するためには人手による補助が必要である．さらに，この手法を用いて生成したラッパは読み込んだ設計記述に対してのみ適用可能であり，同じプロトコルを使う別の設計記述に対して再利用することができない場合があるといった問題点がある．

以上のように，多くのラッパ生成手法では，プロトコルの仕様をFSMの形で与える必要がある．その一方で，多くのプロトコルの仕様書では自然言語とタイミングチャートで仕様が提示されている．したがって，既存のラッパ生成手法を適用するためには，仕様書をもとにして人手でFSMを作成する必要がある．

しかし，仕様書をもとにしてプロトコルの仕様を表すFSMを作成することは簡単ではない．タイミングチャートは，プロトコル上のいくつかの代表的な動作の入出力波形を記述したものである．このため，それぞれの動作例の間の関係は自然言語による注釈を参考にして設計者が推測しなければならない．さらに，それぞれの図を分析する際には，設計者は入出力波形と自然言語による注釈をもとにしてプロトコルの制御手順を推測しなければならない．そこで，本研究ではプロトコルの仕様書をもとにして簡単に仕様記述を作成できるプロトコル表現方法を提案する．加えて，提案表現法を入力とするラッパの自動生成方法を提案する．

プロトコルの制御手順をFSMで表現する場合，極めて多くの種類の制御手順を表現することが可能であるが，表現できる全ての制御手順のうち，実用的なプロトコルを表現する際に必要なものはわずかである．また，どのようなFSMを作成するべきかの判断は全て設計者が行わなければならない．提案表現法では，扱う制御手順を実用的なプロトコルの中で現れるものに限定することで制御手順を簡潔に表現することを可能にする．設計者は，予め定義されている特徴点をタイミングチャート上で探し出して，それを,4種類の宣言文を用いて記述することでプロトコルの制御手順を表現することができる．

以降，第3.章ではプロトコルの表現方法について，第4.章では提案表現法に基づくラッパの自動生成方法について説明する．さらに，第5.章で提案手法に基づいてラッパ生成実験を行った結果を示し，第6.章で本研究をまとめる．

3. 提案するプロトコル表現方法

3.1 本研究で扱うプロトコル

提案表現法の説明に先立って，提案表現法で扱う対象とするプロトコルが満たすべき制限と以降の説明で用いる用語の説明を行う．提案表現法で扱うプロトコルは，クロックに同期してデータをやり取りするメモリマップ型のパラレル通信プロトコルである．シリアル通信のようにクロックごとにデータの断片が渡されるプロトコルは対象としない．

以降の説明では，読み書きなどリクエストを発行するインターフェースをマスタと呼び，リクエストを受け取ってレスポンスを返すインターフェースをスレーブと呼ぶ．また，リクエストを発行する手順によって，プロトコルの形式をブロッキング型とパイプライン型に分類する．ブロッキングプロトコルとは，スレーブが対応するレスポンスを返すまでマスタが次のリクエストを発行できないプロトコルである．これに対しパイプラインプロトコルではリクエストの処理がレスポンスの処理と独立しており，スレーブが許可すれば，マスタはレスポンスが返る前に次のリクエストを発行することができる．

以降の説明では，読み出し動作・書き込み動作などプロトコル上での一連の動作のことを指してシーケンスと呼ぶ．また，何の転送も行っていない状態のことをIDLE状態と呼ぶ．すなわち，シーケンスはIDLE状態から他の状態に遷移することで開始し，IDLE状態に戻ることで終了する．プロトコルは，単一ないし複数のシーケンスから構成され，多くの場合，1つのシーケンスは仕様書のタイミングチャート1枚に相当する．また，本研究ではインターフェースの信号線を制御線とデータ線に分類して扱う．データ線とはアドレスやデータなど送受信されるデータそのものを伝達する信号線のことであり，制御線とはデータのやり取りの手順を制御する信号線のことである．

3.2 提案表現法の説明

提案表現法では，入出力信号リスト，データ線属性，信号エンコーティング，信号間タイミングの4つの領域からプロトコルの仕様記述を作成する．まず、各領域の概要を説明する。

- 入出力信号リストの領域では，設計記述の入出力信号線の中からラッパ生成の対象とする信号線を指定し，指定した信号線を後述するいくつかの観点から分類する．後の3つの領域では，ラッパ生成の対象に指定された信号線の情報を列挙する．このようにして，信号線の結線に関する仕様と信号線上でのやり取りに関する仕様を切り分けて扱う．
- データ線の属性の領域ではデータ線に対して，信号エンコーディングの領域では制御線に対して詳細な仕様を決定する．このようにして，制御線に関する仕様とデータ線に関する仕様を切り分けて扱う．
- 信号間のタイミングに関する仕様は信号間タイミングの領域が取り扱い，信号エンコーディングの領域では，信号線間の結線や信号線の論理値といった時間の概念を含まない仕様を取り扱うこのようにして，タイミングに関する仕様のみを他の仕様から切り分けて扱う．

以降に，それぞれの領域について詳しく説明する．また，第3.3節では実在するプロトコルに提案表現法を適用して，仕様記述を作成する手順を説明する．

まず，入出力信号リストの領域について説明する．ここでは，ラッパ生成の対象とする信号線に対し，信号名，信号のビット幅，制御線とデータ線の区別，信号のドライバの情報を列挙する．信号のドライバとは，信号線に出力するインターフェースのことで，マスタ，スレーブ，外部のモジュールのいずれかである．また，クロックの信号線とリセットの信号線を他の制御線と区別する．設計者は，プロトコルの仕様書に記載されたインターフェースの信号線の一覧を参照して，入出力信号リストの仕様記述を作成する．

次に，データ線属性の領域について説明する．ここでは，データ線に分類された信号を，「信号線 ADDR はアドレスを伝達する」のように伝達される情報の種類によって分類する．異なるプロトコルのデータ線同士を接続する際にはここでの分類を用いてデータ線同士の対応をとる．

信号間タイミングの領域では，1つのシーケンス内の制御線と制御線，制御線とデータ線のタイミング関係を4種類の宣言文を組み合わせて表現する．ここで用いる4種類の宣言文では，全ての信号は有効値か無効値の2種類の値をとるものとして扱う．データ線においては，有効値はデータ線上に有効なデータが出力されていることに，無効値は出力されていないことに相当する．これに対し，制御線に対しては，有効値は信号線上でHighなのかLowなのかを制御線ごとに定義する必要がある．また，多ビット幅の制御線を扱う際には，制御線の意味をふまえて1ビットの制御線複数本にデコードした後に，4種類の宣言文を適用する．なお，有効値・無効値の定義，デコードの規則は次に説明する信号エンコーディングの領域で取り扱う．

以下に，信号間タイミングの領域で用いられる4種の宣言文について説明する．これらの宣言文は，信号名とサイクル単位の時間を引数として取る．以降の説明では，引数型 sig は信号名，引数型 time は時間を表す．

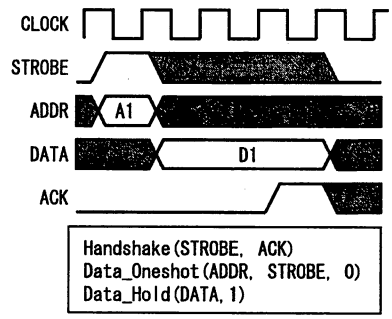

図1 例題シーケンス
Fig. 1 An example sequence

- Handshake(sig sigStart, sig sigEnd),
- OverlapHandshake(sig sigStart, sig sigEnd) では，sigStart が有効値をとることで開始して，sigEnd が有効値をとることで終了する期間を表現する．この期間を以降の説明ではハンドシェーク期間と呼ぶ．通常，Handshake() 文は sigStart のドライバと sigEnd のドライバに異なるものを指定して，プロトコル上の待機機構を表現するために使用される．なお，ハンドシェーク期間中では sigStart の値は読み込まれず，ハンドシェーク期間外では sigEnd の値は読み込まれないとする．設計者は，ハンドシェーク期間が終了した後に sigStart を読み込み始めるタイミングの違いによって，Handshake() 文と OverlapHandshake() 文を使い分ける．Handshake() 文では sigEnd が有効値をとったサイクルの次のサイクルから sigStart を読み込み，OverlapHandshake() 文では sigEnd が有効値をとったサイクルから sigStart を読み込む．
- Data_Hold(sig sigTarget, time latency) では，Handshake() 文，OverlapHandshake() 文で定義されたハンドシェーク期間中に有効値をとり続ける信号線を表現する．latency は，sigStart が有効値をとってから sigTarget が有効値をとるまでの遅れをサイクル単位で表したものである．
- Data_Oneshot(sig sigTarget, sig sigTrigger, time latency) では，1サイクルの間有効値をとる信号線 sigTarget を表現する．latency は，sigTrigger が有効値をとってから sigTarget が有効値をとるまでの遅れをサイクル単位で表したものである．sigTrigger とする信号は，Handshake() 文あるいは OverlapHandshake() 文が宣言されている場合は sigStart か sigEnd から選ぶ．どちらも宣言されていない場合は，基準となる信号 signalA について Data_Oneshot(signalA, signalA, 0) と宣言し，残りの信号のタイミングを signalA を基準として宣言する．

ここで，図1に示したタイミングチャートを用いて4種の宣言文の説明を補足する．信号 STROBE と信号 ACK は1ビット幅の制御線であるとして，信号 ADDR と信号 DATA は多ビット幅のデータ線であるとする．図1では，1ビット信号線

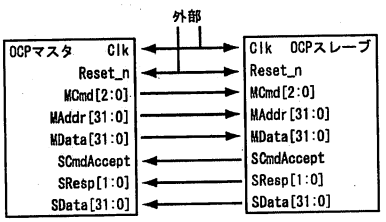

図 2 例題で用いる OCP インターフェース

Fig. 2 OCP interface used in the example

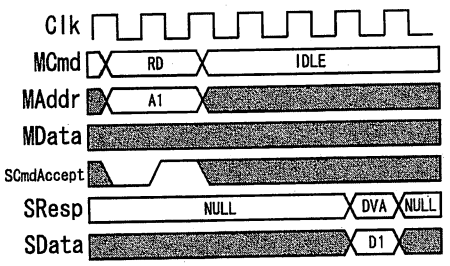

図 3 OCP 読み出しシーケンスのタイミングチャート

Fig. 3 Timechart for the OCP read sequence

に対して，上線は有効値，下線は無効値，網掛け部は値が参照されないことを示す．また，多ビット線に対しては，白抜き部は有効値，網掛け部は無効値を示す．信号間タイミングの仕様記述を作成する際には，最初にシーケンスの開始と終了を定義する制御線信号を見つけ出して Handshake() 文もしくは OverlapHandshake() 文で表現する．その後に残りの信号のタイミングをこれらの信号を基準として表現する．図 1 では，信号 STROBE と信号 ACK がシーケンスの開始と終了を定義している．さらに，信号 ACK が有効値をとるとき信号 STROBE は参照されていないので，両者の関係を Handshake(STROBE, ACK) と記述する．信号 ADDR は信号 STROBE が有効値をとると同時に有効値となり，1 サイクルだけ有効値を保つので，Data_Oneshot (ADDR, STROBE, 0) と記述する．信号 DATA は信号 STROBE が有効値をとった次のサイクルから有効値をとり，ハンドシェーク期間中有効値を保ち続けるので，Data_Hold (DATA, 1) と記述する．

対象とするプロトコルがブロッキングプロトコルの場合は，各シーケンスに対し以上のようにタイミング仕様を記述する．また，パイプラインプロトコルのタイミング仕様を表現する場合は，リクエスト処理に関するタイミング仕様とレスポンス処理に関するタイミング仕様を別々に記述する．

信号エンコーディングの領域では，固有プロトコルの制御線に関して 3 種類の情報を取り扱う．まず，プロトコル仕様書に記載された信号線の説明を参照して，制御線ごとに有効値と実際の論理値の対応を記述する．また，多ビットの制御線をデコードする場合は，その変換規則を記述する．さらに，複数のシーケンスから実行するシーケンスを選択する規則を記述する．

3.3 仕様記述作成の例題

この節では，OCP (Open Core Protocol) [4] の読み出し動作のシーケンスを例題として，提案表現法で仕様記述を作成する手順を示す．例題として用いる OCP インターフェースを図 2 に，読み出し動作時のタイミングチャートを図 3 に示す．

表 1 OCP の入出力信号リストの仕様記述

Table 1 Input/Output signals specification for OCP

信号名	ビット幅	ドライバ	信号線種別
Clk	1	外部	クロック
Reset_n	1	外部	リセット
MCmd	3	マスタ	制御
MAddr	32	マスタ	データ
MData	32	マスタ	データ
SCmdAccept	1	スレーブ	制御
SResp	2	スレーブ	制御
SData	32	スレーブ	データ

```
リクエスト処理
Handshake(ocpMCmd_1, ocpSCmdAccept)
Data_Hold(ocpMCmd_1, 0)
Data_Hold(ocpMAddr, 0)
レスポンス処理
Data_Oneshot(ocpSResp_1, ocpSResp_1, 0)
Data_Oneshot(ocpSData, ocpSResp_1, 0)
```

図 4 OCP 読み出しシーケンスのタイミング仕様記述

Fig. 4 Timing specification for the OCP read sequence in the proposed protocol description language

まず，プロトコル仕様書に記載された信号線の定義一覧をもとに入出力信号リストの仕様記述を作成する．図 2 に示したインターフェースに対する仕様記述を表 1 に示す．さらに，データ線の伝達する情報について仕様書を調べると，信号線 MAddr が伝達するのはアドレス，信号線 MData・SData が伝達するのはデータであることが分かる．ここからデータ線属性の仕様記述を作成する．

次に，多ビットの制御線に着目する．信号線 MCmd は 3 ビット幅の，信号線 SResp は 2 ビット幅の制御線である．仕様書を参照してこれらの信号線の詳しい説明を調べると，インターフェースは信号線 MCmd の値が RD である時に読み出し動作を開始し，信号線 SResp の値が NULL 以外の時にレスポンスの動作を開始すると記載されている．そこで，設計者は信号線 MCmd の値が RD の時に有効値をとる信号線 MCmd_1 と，信号線 SResp の値が NULL 以外の時に有効値をとる信号線 SResp_1 を定義する．多ビット制御線の扱いが決定した後で，信号エンコーディングの仕様記述と信号間タイミングの仕様記述の作成を行う．

先に信号エンコーディングの仕様記述の作成手順について説明する．まず，仕様書を参照して制御線の有効値を調べると，信号線 Reset_n の有効値は Low で，それ以外の制御線の有効値は High であることが分かる．ここから，1 ビット制御線に対する信号エンコーディングの仕様記述を作成する．さらに，

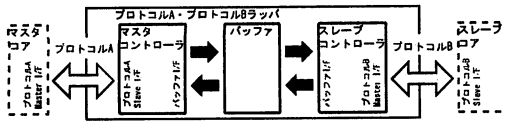

図 5 提案するラッパ構造図

Fig. 5 Block diagram of the proposed protocol wrapper

多ビット制御線の変換規則を仕様記述に加える．また，仕様書によると，読み出しシーケンスが選択されるのは MCmd の値が RD である時なので，この選択規則を仕様記述に加える．

最後に，前節で説明した手順に従って，信号間タイミングの仕様記述を作成する．OCP はリクエストとレスポンスの処理が分離しているパイプラインプロトコルである．仕様書と図 3 から，リクエストに関係している信号線は MCmd，MAddr，MData，SCmdAccept であり，レスポンスに関係している信号線は SResp と SData であることがわかる．読み出しリクエストの処理は信号 MCmd_1 が有効値をとることで開始し，信号 SCmdAccept が有効値をとることで終了する．また，リクエスト処理の期間中は，信号 MCmd，MAddr，MData の値は変化しない．レスポンスの処理は信号 SResp_1 が有効値をとることで開始するが，終了する信号は定義されていない．また，SResp_1 が有効値をとると同時に SData が有効値をとる．以上の分析より，信号間タイミングの仕様記述は図 4 のように記述される．

4. 提案するラッパ生成手法

4.1 ラッパ全体の構造

本研究では，提案表現法をもとにして，図 5 に示す構造のラッパを生成する．ラッパは，マスタの出力するリクエストを受け取るマスタコントローラと，スレーブにリクエストを出力するスレーブコントローラ，両者を接続するバッファから構成される．以降の説明では，マスタコントローラとスレーブコントローラを総称してラッパコントローラと呼び，IP 固有インターフェースプロトコルを省略して固有プロトコルと呼ぶ．ライブラリベース手法に基づいて考えると，ラッパコントローラは固有プロトコルをバッファの読み書きという内部共通プロトコルに変換するラッパであると考えられる．また，本研究で用いたバッファは，固有プロトコルのデータ線の情報を伝達する FIFO レジスタと，ラッパコントローラの制御情報をやり取りする FIFO レジスタから構成される．それぞれの FIFO レジスタは，データ属性ごとや制御情報の種類ごとにさらに細かく分かれている．

ラッパコントローラは，提案表現法で記されたプロトコルの仕様記述をもとに，次節で説明する手順によって生成される．また，既存のラッパコントローラが利用可能な場合は既存のものを再利用することによって，ラッパ生成作業における新規設計の割合を減らすことができる．

4.2 ラッパコントローラ生成手法

図 6 にラッパコントローラの構造図を示す．マスタコントローラ・スレーブコントローラともに同じ構造を持つ．ラッパ

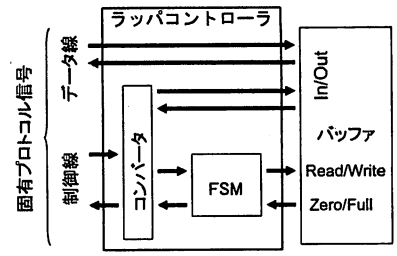

図 6 ラッパコントローラ構造図

Fig. 6 Block diagram of the wrapper-controller

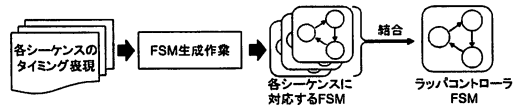

図 7 ラッパコントローラ FSM 生成作業フロー

Fig. 7 Synthesis flow for FSM of the wrapper-controller

コントローラは，バッファの読み書き・信号の入出力タイミングを制御するラッパコントローラ FSM と，固有プロトコルの制御線信号をラッパコントローラ FSM の入出力信号に変換するコンバーターから構成される．コンバーターは純粋な組み合わせ回路であり，記憶素子を含まない．この構成は，プロトコル表現が信号間タイミングと信号エンコーディングに分けて記述されることと対応している．

ラッパコントローラの設計記述において，入出力信号リストの仕様記述は，ラッパコントローラの入出力信号を決定するために使用される．生成結果を HDL（ハードウェア記述言語）などで出力する際に，設計記述の入出力宣言部はラッパコントローラの入出力信号リストから生成される．データ線属性の仕様記述は，固有プロトコルのデータ線とバッファのデータ入出力信号線を接続する際にデータ属性ごとの対応をとるために使用される．

コンバーターは，信号エンコーディングの仕様記述をもとにして生成する．有効値・無効値と信号線上での High・Low の対応，多ビット信号線のデコードの規則，シーケンス選択の規則は，固有プロトコル制御線とラッパコントローラ FSM 入出力信号線の間の真理値表と考えることができる．この真理値表から組み合わせ回路を合成することで，コンバータを生成することができる．

ラッパコントローラ FSM は，図 7 に示した手順によって，信号間タイミングの仕様記述をもとにして生成する．手法の入力として与えられるのは，各シーケンスごとのタイミング表現である．それぞれのタイミング表現に対して FSM 生成作業を行い，各シーケンスに対応する FSM を生成する．タイミング表現から FSM を生成する作業は，FSM テンプレートをタイミング表現に従って加工することによって行われる．この時使用される FSM テンプレートはラッパコントローラとプロトコルの種類によって選択される．ラッパコントローラ FSM 生成手法では，各シーケンスに対応する FSM が生成された後にそれ

らを結合してラッパコントローラ FSM を生成する．各シーケンスは IDLE 状態から動作を開始して，IDLE 状態に戻ることで動作を終了するので，各シーケンスに対応する FSM の間で IDLE 状態を共有することによって FSM を結合することができる．

こうして生成された入出力信号線，データ線とバッファの結線，コンバーター，ラッパコントローラ FSM の設計記述を結合してラッパコントローラの設計記述を生成する．

5. 実　　験

5.1 ラッパコントローラ生成ツールの実装

実験を行うにあたって，ラッパコントローラの入出力信号線の Verilog 記述を生成する処理とラッパコントローラ FSM の Verilog 記述を生成する処理を C++を用いて実装した．本実験で実装した FSM 生成ツールは，マスタコントローラのラッパコントローラ FSM を生成することができる．

5.2 実験手順

実験で用いたプロトコルは，AMBA-AHB [3] プロトコルと OCP [4] である．以降の説明では AMBA-AHB を省略して，AHB と呼ぶことにする．実験では，OCP マスタコントローラ，OCP スレーブコントローラ，AHB マスタコントローラの 3 つをラッパコントローラ生成手法に基づいて生成し，それらを組み合わせて OCP→OCP ラッパと AHB→OCP ラッパを生成した．生成したラッパは各プロトコルでの 1 ワード単位の読み出し動作と書き込み動作を扱う．さらに，生成したラッパに対し，シミュレーションによる動作確認と形式的検証ツールによる動作確認を行った．シミュレーションによる動作確認では主に，ラッパを経由してリクエストの種類やデータ線の情報が正しく伝達されていることを確認し，形式的検証ツールによる動作確認では主に，ラッパの制御線がプロトコルの正しい制御手順に従って遷移していることを確認した．

5.3 実験結果

プロトコル仕様記述の作成作業では，全てのプロトコルにおいて仕様記述を作成することができた．また，生成されたラッパに対する動作確認においては，シミュレーションによる動作確認と形式的検証ツールによる動作確認の両方でラッパが正常に動作することが確認できた．

提案表現法とラッパ生成手法の導入による生産性の向上を図る指標として，提案表現法で記述したプロトコルの仕様記述の行数と，生成したラッパの Verilog 記述の行数を比較することにする．表 2 にラッパコントローラ単体を生成した際の記述行数の比較結果を示し，表 3 にラッパ全体における記述行数の比較結果を示す．なお，記述行数を計測する際には，空白行やコメント行はすべて計測から除外した．また，ラッパ記述の総行数には，2 つのラッパコントローラの行数の他に，バッファの設計記述やラッパコントローラとバッファの接続部の設計記述の行数が含まれている．表 2 を参照すると，ラッパコントローラを生成する際に提案表現法による仕様記述の 4 ～ 5 倍の行数の Verilog 記述が生成されることが見て取れる．また，表 3 を参照すると，提案表現法の 7 倍程度の行数のラッパ設計記述が生成されたことがわかる．

表 2　提案表現法での仕様記述の行数と Verilog でのラッパコントローラ設計記述の行数の比較
Table 2　Number of lines for protocol specification in proposed description language and wrapper-controller design in Verilog

ラッパコントローラ	仕様記述	設計記述
AHB マスタ	48	195
OCP マスタ	32	189
OCP スレーブ	34	157

表 3　提案表現法での仕様記述行数と Verilog でのラッパ記述の総行数の比較
Table 3　Number of lines for protocol specification in proposed description language and wrapper design in Verilog

ラッパ名	仕様記述	ラッパ記述
AHB → OCP ラッパ	80	481
OCP → OCP ラッパ	66	458

6. 本研究のまとめ

本研究では，プロトコル仕様書の自然言語およびタイミングチャートからプロトコルの仕様記述を生成できるプロトコル仕様表現方法と，それを入力とするラッパ生成手法を提案した．

提案表現法では，プロトコルの仕様を入出力信号リスト，データ線属性，信号エンコーディング，信号間タイミングの 4 つの領域に分けて記述する．この表現法を用いることで，信号線のタイミング関係と，信号線上での実際の論理値を切り分けて扱うことができる．信号間タイミングの仕様記述を作成するにあたっては，予め指定された特徴点をタイミングチャートから探し出して，4 種類の宣言文を用いて列挙することにより簡単に仕様記述が作成できる．また，提案するラッパ生成手法は，プロトコル仕様記述をもとにテンプレートを加工することにより，提案表現法によるプロトコル仕様記述から機械的にラッパを生成することができる．実験では，実際に AMBA-AHB プロトコルと OCP の間を変換するラッパを生成し，ラッパが正しく作られることを確認した．また，提案表現法からラッパを生成することで，設計者が記述するべき行数を大幅に削減できることを確認した．

文　　献

[1] Roberto Passerone, James A. Rowson, Alberto Sangiovanni-Vincentelli "Automatic Synthesis of Interfaces between Incompatible Protocols" DAC'98 SanFrancisco, California

[2] James Smith, Giovanni De Micheli "Automated Composition of Hardware Components" DAC'98 SanFrancisco, California

[3] AMBA Specification 2.0 (http://www.arm.com/products/solutions/AMBA_Spec.html)

[4] OpenCoreProtocol Specification ver 2.11 (http://www.ocpip.org/socket/ocpspec/)