C_008

Rip-up IPを用いたカスタマイズ設計環境

IP re-used design environment for quick customization using a Rip-up IP

亀井　智紀†　　渡邊　孝博†
Tomoki Kamei　　Takahiro Watanabe

1．はじめに

システムLSIの開発では、対象となるシステムが年々大規模化していく傾向が続き、増大する開発期間と開発コストに対してどのように設計生産性を向上させるかが問題となっている。この解決法の一つとして、IP（Intellectual Property）を利用した設計が近年注目されている。IPとは、あらかじめ設計され性能や動作が保証された再利用可能な回路ブロックである。これを設計の一部として用いることで設計の手間を減らし、生産性を向上させることができる。

そこで、我々の研究グループで提案しているRip-up IP機能を含めて、IP再利用設計環境を考察する。Rip-up IPをカスタマイズする設計環境はすでに試作し、IP評価に用いているが、ターゲットとするIPが特定され、他のIPへの適用が困難であった。そこで今回、Rip-up方式で設計された全てのIPを処理できる設計環境を開発した。

2．Rip-up　IP方式

IPを利用してSoC設計を行う場合、目的とするシステムに完全に一致するIPは存在し得ない。そのため、IPをそのままシステムに組み込むと、少なからず不要な機能を含んだまま実装することになり、その分回路規模が大きくなる。

この問題の解決策として、我々はRip-up IPを提案している。これはカスタマイズできるIPであり、ソフトIPでの提供を目的として上記の問題を解決するものである。つまり、IPの中身を機能ごとにモジュール化し、不要な機能は容易に削除できる。さらに新たな機能を追加することで目的のシステムに必要な機能を達成する。Rip-up IPではIPの全ての機能を組み込んだ場合は、従来のIPより回路規模が増大するが、不要な機能を削除することにより、回路規模が縮小され、動作速度の向上を期待することができる。また、限られたチップサイズ上では、削除された分だけ新たな機能に使用することができる。

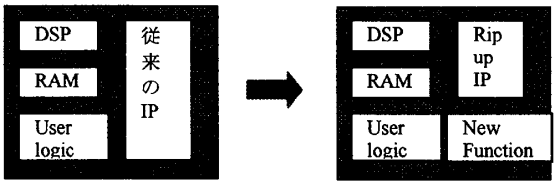

図1．Rip-up IPを用いたSoC設計

† 早稲田大学大学院情報生産システム研究科

3．従来のRip-up IP開発環境

Rip-up IPをカスタマイズするための設計環境として我々は図2に示すツールを開発し、使用してきた。このツールの特徴と問題点は次の通りである。

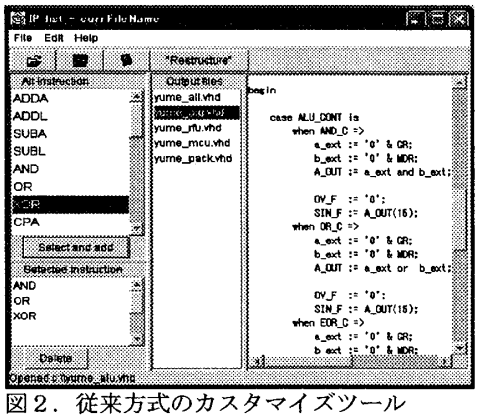

図2．従来方式のカスタマイズツール

(1) カスタマイズの実現方法

Rip-up IPとして現在のところ2つのマイクロプロセッサの設計データを有している。カスタマイズ（機能の削除）の方法は、あらかじめツールの内部に保持しているIPの機能ごとのプログラムソース(HDL記述)のうち、不要と指定されたものは出力に組み込まないことで実現している。つまり、機能を削除するといっても実際にIPのプログラムソースから削除するわけではなく、削除指定されていないモジュールのみを連結することによって、結果的に指定された機能を取り除いたIPが得られる仕組みである。

(2) 問題点

問題点として、あらかじめカスタマイズされるIPのプログラムソースを機能ごとに分割して登録しておかなければ使用できないことが挙げられる。

図2の左側のテキストボックスには、使用可能な全命令が表示され、右側のテキストボックスにはカスタマイズされたIPのソースコードが表示される。これらは、ツール自身があらかじめ抱え持っているデータを適宜組み合わせて、表示しているにすぎない。つまり、新たにIPを作成してカスタマイズしようとすれば

- IPのHDLプログラムを解析する
- 解析結果からカスタマイズ可能な部分を機能ごとに抽出する
- 一方でカスタマイズツールのプログラム(Java)も解析しながら、機能ごとに抽出したIPのプログラムをツールに取り込む

という作業を行わなければならない。そのためには、VHDL等のハードウェア記述言語の習得に加え、ツール記述言語であるJavaの知識も必要になる。すなわち、本環境

のユーザは IP の知識と HDL 設計技術に加えて、Java のスキルが必要となり、効率的な設計環境という点で問題があった。そのような理由もあり、本ツールは現在のところ 1 種類の Rip-up IP にしか対応しておらず、他の IP への適用も進んでいない。

4. 今回開発したカスタマイズツール

カスタマイズツールの内部にターゲット IP のプログラムを持たせる手法は、前述のように拡張性や作業の容易性という点で不利である。

そこでターゲットプログラムをツールから独立させ、利用の都度、外部から読み込ませることのできるツールを開発した。この手法は IP のプログラムをツール内部に持たないため、適用 IP が限定されないという利点がある反面、以下の点を考慮しなければならない。

(1) カスタマイズするためには、ユーザに機能（命令）を提示して必要･不要を選択してもらう必要があるがどうやって初めて読む IP の機能を抽出するか

(2) 機能を抽出して、不要な命令を指定したとして、その命令に対する処理の記述は IP のプログラム上のどこに該当するのか

4.1 新ツールの技法上の特長

(1) IP の機能（命令）表示法

我々の研究グループで開発した 8086 互換プロセッサの Rip-up IP を例に挙げ、新ツールでの使用命令の表示方法を示す。この IP は VHDL により記述されており、変数の宣言を行うパッケージファイルが存在する。このパッケージファイルの中に constant 宣言されている命令語があるため、これらの字句解析を行うことで必要な命令語を抽出する。ファイル内には命令語以外の変数も宣言されているので、図３のように From-To をコメントで挿入することにより抽出範囲を明示する。

VHDL のソースリストからトークンに分解するために Java の StreamTokenizer クラスを使用した。このクラスは入力ストリームの字句解析を行う。この結果、図４の左側のテキストボックスに命令語が表示可能となり、ユーザはここで不要な命令を選択する。

(2) 削除範囲の決定法

Rip-up IP の特長として、不要な機能を削除することが前提であるので、機能に対応する削除範囲を見つけることは容易である。VHDL で作成された IP であれば、case 文中の when 文節中に対応する処理は集約される。case 文のネストやコメント文の考慮等の必要性はあるが、上述の StreamTokenizer クラスと文字列操作で削除範囲は決定可能になる。また、J2SE1.4 より Java でも正規表現がサポートされたため、目的の文字列検索に利用している。

4.2 動作検証

開発した新ツールに従来ツールで対応している Rip-up IP を読み込ませ、カスタマイズ作業を行った結果、従来ツールと同等の結果を出力した。内部にターゲットプログラムを持たせなくても Rip-up IP のカスタマイズは可能であることを確認した。

さらに、8086 互換プロセッサに関しては、命令数も多いため従来ツールでは対応していなかったが、今回のツールによってカスタマイズが可能になった。

```
-- Instruction start
constant PUSH    : std_logic_vector(7 downto 0) := "11111111";
constant POP     : std_logic_vector(7 downto 0) := "10001111";
constant POPreg  : std_logic_vector(4 downto 0) := "01011";
constant XCHG    : std_logic_vector(6 downto 0) := "1000011";
constant XCHGax  : std_logic_vector(4 downto 0) := "10010";
constant INN     : std_logic_vector(6 downto 0) := "1110010";
constant INdx    : std_logic_vector(6 downto 0) := "1110110";
constant OUTT    : std_logic_vector(6 downto 0) := "1110011";
constant OUTdx   : std_logic_vector(6 downto 0) := "1110111";
-- Instruction end
```

図３．コメント指定した VHDL リスト

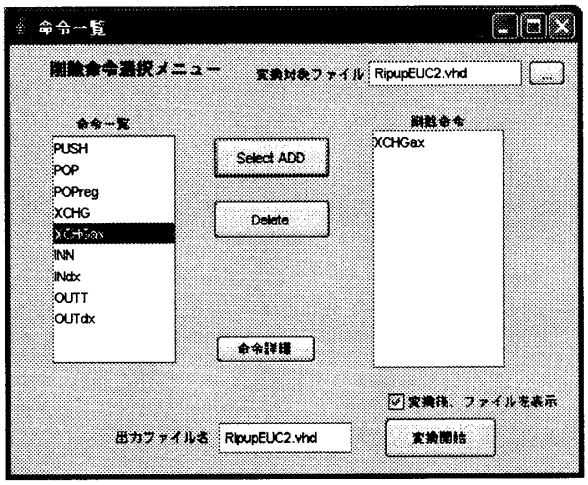

図４．字句解析によりリストから命令語を抽出

5．おわりに

本研究では、FPGA に実装する IP の再利用の設計環境開発を行った。この設計環境は Rip-up IP 方式を利用するものである。従来の設計環境は、ターゲットとする IP をツール自身の中に取り込む必要があるため、新たな IP への適用が困難であった。今回作成したツールは Rip-up 方式の設計記述である IP ならば、全てカスタマイズ可能となる。

今後は、Verilog-HDL や SystemC で作成された IP にも対応させていく予定である。

参考文献

[1]柴田智彦：IP 再利用の設計環境に関する研究、山口大学大学院理工学研究科修士論文、2003 年 2 月。

[2]野村知弘：マイクロプロセッサの効率的な設計手法に関する研究、山口大学大学院理工学研究科修士論文、2005 年 2 月。

[3]徐軼韜：FPGA-IP 利用の一手法とその設計環境、早稲田大学大学院情報生産システム研究科修士論文、2005 年 1 月。

[4]岩谷宏：Java によるテキスト処理入門、ソフトバンクパブリッシング株式会社、2002 年。