L_035

# Webベース共同プロセッサ開発環境

## Web-based Collaborative Processors Development Environment

柳澤 秀明† 松本 勝慶† 上原 稔† 森 秀樹†
Hideaki Yanagisawa, Katsuyoshi Matsumoto, Minoru Uehara, Hideki Mori

### 1. はじめに

我々は、ISA（Instruction Set Architecture）ベースでのプロセッサ開発環境（PDE：Processor Development Environment）C-DASH(C-like Design Automation Shell)[1][2]の開発を進めている。プロセッサを短期間に開発するためには、チームで協力して開発を進める必要があるが、ローカル環境で開発を進める場合、プロセッサ開発に必要なツールを開発者それぞれのローカルホストにインストールする必要がある。また、開発者が、独自に開発を進めてしまい、既に編集中のファイルに対して、他の開発者が変更を行ってしまうなど、ファイルの整合性が問題となる。

チームで効率的に開発を進めるためには、資源を集中管理できる共同開発環境（CDE：Collaborative Development Environment）が必要である。しかしながら、PDEとしてCHESS/CHECKERS（ベルギーTarget Compiler Technologies N.V.）[3]、Xtensa（米Tensilica）[4]、ASIP Meister（ASIP Solutions）[5]、LISATek（米CoWare）[6]など、様々な研究開発が行われているが、プロセッサ開発のためのCDEについての記述を見つけることは出来ない。

本論文では、チームで協力して、プロセッサ開発を行うための Web ベース共同プロセッサ開発環境（WCPDE：Web-based Collaborative Processor Development Environment）について述べる。開発環境を Web ベースとすることで、開発者のプラットフォームに依存することなく、世界中のどこからでも開発に参加することができるようになる。また、ファイルを 1 箇所で管理できるため整合性の問題を解消できる。

### 2. 共同開発環境

チームで共同開発を行う場合、ソースコードのバージョン管理としてCVS(Concurrent Version System)[7][7]は、広く利用されている。しかしながら、CVSでは、ファイル管理を行うだけであり、CDEとして利用するためには、開発に必要なツールを開発者が用意しなければならない。また、共同開発をスムーズに進めるためには、情報交換のためにメーリングリストや掲示板などのコミュニケーションツールが必要となる。

オープンソースプロジェクトでは、SF.net(SourceForge.net)[8][8]などのCDEを利用した開発も行われている。SF.netでは、CVSリポジトリ、メーリングリスト、掲示板、バグ追跡システム、Webサイトホスティング、リソースファイルのダウンロードサービス、シェル環境、コンパイル・ファーム、プロジェクト管理機能などをWebベースで提供している。しかしながら、CVSクライアントのインストールが必要となる。また、汎用的なCDEであるため、特定の用途に特化したCDEとはならない。

本研究では、SF.net の機能を参考にし、プロセッサ開発に特化した WCPDE の構築を行うことを目的とする。

### 3. C-DASH (C-like Design Automation Shell)

プロセッサを開発するためには、HW（Hardware）の設計のみならず、シミュレータ、アセンブラ、逆アセンブラ、コンパイラなどの SWDE（Software Development Environment）の開発も必要となる。

このため、我々は、ISA ベースでのプロセッサ定義を可能とする PDE、C-DASH の開発を行っている。C-DASH では、ISA ベースでのプロセッサから以下の生成を行うことが出来る。

- Instruction set level simulator (Java or C)
  CUI または、GUI を持ったインタプリタ型 ISA レベルシミュレータ。
- Simulation Compiler
  コンパイル型シミュレーションを可能にするためにターゲットマシンの実行コードをホストマシンの実行コードに変換する。
- Hardware simulator (SpecC)
  C-DASH は、SpecC でのハードウェアシミュレータの生成も行う。このシミュレータは、プロセッサのアーキテクチャのチェックを行うために利用する。
- Compiler (GCC)
  C-DASHでは、GCC（GNU Compiler Collection）[9]をクロスコンパイラとして利用するために，GCCのマシーン定義ファイルを自動生する。（現状では、RISC、CISC型プロセッサに対する、命令セットの動作をRTL記述出力するのみ）
- Assembler
  アセンブリ言語で記述されたプログラムをオブジェクトコードに変換する。
- Disassembler
  オブジェクトコードをアセンブリ言語に変換する。デバッグやアプリケーションプログラムから命令セットの抽出などに利用する。
- HDL description (Verilog-HDL)
  C-DASH は、HW 記述として Verilog-HDL 記述を生成する。C-DASH から生成されるプロセッサの HDL 記述は、論理合成可能な記述である。

C-DASH では、単一のプロセッサ記述から、HW/SWDE を自動生成することができるためプロセッサ設計者の開発負担を減らすことができる。また、ISA でプロセッサを定義しているため、命令セットの変更を簡単に行うことができる。

†東洋大学工学部情報工学科, Toyo University

### 3.1. 記述レベル

C-DASH では、ビヘイビア記述、クロックベース記述、パイプライン記述の 3 つの記述レベルを持つ。

- ビヘイビア記述
  C 言語を使って、自由に動作記述を行うことができる。この記述では、HW の生成は行わず、SWDE の生成のみを行う。
- クロックベース記述
  クロックに基づく動作記述を行う。SWDE の生成と HW として固定された、4 段ステージ（フェッチ、デコード、実行、ライトバック）での HDL 記述の生成を行う。（1 クロックでの動作に限定）
- パイプライン記述
  プロセッサのパイプラインステージを定義する。HW と SWDE の生成を行う。

### 3.2. プロセッサ定義

C-DASHで、プロセッサを定義するためには、リソース（メモリ、レジスタなど）定義と命令の動作定義が必要である。ビヘイビア記述でのプロセッサ定義例を図 1図 1 命令定義示す。

```
reg   gr   16   8;
reg   pc   16;
ram   m   16   65536;
instructin   m   pc;
instruct adda {00100000ddddssssiiiiiiiiiiiiiiii} {
   gr[d] = gr[d] + m[i];
}
asm adda {<label><opcode><reg>,<label.address>} {
   d = number($3);
   s = 0;
   i = address($5);
}
asmgroup adda {suba, or, ...}
```

図 1　命令定義

C-DASH では、**"reg"** でレジスタのビット幅や個数を定義する。また、**"ram"**によりメモリのビット幅とサイズを定義し、命令メモリとプログラムカウンタを**"instructin"**により明示する。

各命令の動作は、**"instruct"**命令により定義する。C-DASH では、ニーモニック名（adda）、ビットパターン（0010・・・・）と命令の動作を定義し、命令のフォーマットを**"asm"**命令により定義する。"<label>"は、プログラム中で使われるラベルを表し、"<opcode>"でニーモニック名を表す。"<reg>"でオペランドがレジスタであることを表し、"<label.address>"でオペランドが即値または、ラベルであることを表す。**"asm"**中の**"d,s,i"**は、**"instruct"**のサブビット**"dddd"**、**"ssss"**、**"ii…"**をそれぞれ表し、"number()"でレジスタ番号を"address()"で即値やアドレスをサブビットに埋め込むことを表している。**"$N"**（N=3,4…）は、","も含めた番号を表している。

## 4. Web ベース共同プロセッサ開発

C-DASH を利用したプロセッサ開発は、ローカル環境で行われていた。複数人で共同開発を行う場合、開発者間でのファイルの同期が問題となる。

プロセッサ開発を、Web ベース CDE とすることで、ブラウザとテキストエディタさえあれば、どこからでもプロジェクトに参加でき、いつでも、最新リソースにアクセスすることができ、ファイルの同期の問題を解消できる。

このため、Java サーブレット、Ant、CVS を利用し、C-DASH サーバを構築する。C-DASH サーバは、汎用的な CDE と異なり、プロセッサ開発に特化した CDE であり、プロセッサの定義ファイルをアップロードすることで、プロセッサのシミュレータ、アセンブラ、逆アセンブラ、コンパイラ、HDL 記述などを提供を行う。

### 4.1. C-DASH サーバ

C-DASH サーバは、以下の機能で構成する。

- ユーザ管理
  開発者の登録、削除。
- プロジェクト管理
  プロジェクトの作成・削除、メンバー登録・削除、サブプロジェクトの作成・削除など。
- サブプロジェクト
  個人用の開発環境を提供。開発者が自由に利用できるスペース。ファイルの変更などは、他の開発者への影響を与えない。
- バージョン管理
  CVS を利用しソースファイル、ドキュメント、実行ファイルなどの管理を行う。
- アクセスコントロール
  プロジェクトメンバに対するアクセス権限を設定することができる。（参照、変更の許可）
- 情報管理
  プロジェクトの進行状況（開発・更新記録）、バグ情報、バグの修正情報、意見交換（掲示板）
- ライブラリ管理
  IP の登録、利用
- Web-based シミュレーション
  Web 上でのシミュレーションの実行
- リリースシステム
  開発者による命令の追加や、変更がアップロードされたとき、他の開発者の承認を受ける。また、変更により影響が出ないかチェックする。

### 4.2. C-DASH サーバの動作

C-DASH サーバは、開発者によりアップロードされたプロセッサ定義ファイルに基づき HW/SWDE の自動生成を行う。

C-DASH では、SWDE として Java による IDE（Integrated Development Environment）の実装を行っている。このため、C-DASH サーバでは、JWS（Java Web Start）を利用し、IDE のダウンロードおよび、実行を Web ページ上の JNLP（Java Network Launching Protocol）へのリンクをクリック

するだけで、新たに開発した IDE の実行が出来るように自動化している。

開発者のローカル環境に JRE（Java Runtime Environment）がインストールされていれば自動的に IED の実行が可能となる。

サーバの動作を以下に示す。

1. 開発者が Web ブラウザでプロジェクトの Web ページへアクセス。
2. プロセッサ定義ファイルをアップロード。
3. C-DASH サーバは、アップロードされたファイルをサーバ上のプロジェクトディレクトリに保存。
4. プロセッサ定義ファイルを C-DASH の入力とし HW の生成とシミュレータ、アセンブラ、逆アセンブラなどの SWDE のソースファイルの生成を行う。
5. ソースファイルとライブラリなどをコンパイルし、SWDE の class ファイルを生成。
6. 生成された class ファイルを JAR 化。
7. JAR ファイルに署名。

追加／修正内容に他の命令に影響を与える場合などは、BBSを利用しあらかじめ情報の公開を行うことで、開発者間の問題を解消する。サーバの動作を以下に示す。(図 3)

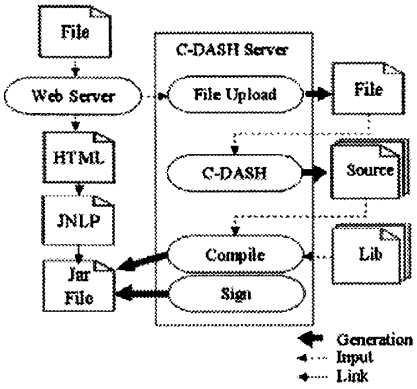


図 3　C-DASH サーバによるシミュレータの生成

以上の処理を行うことで、プロジェクト作成時に生成された JNLP ファイル、HTML ファイルのリンクからシミュレータの起動、プロセッサ定義ファイルのダウンロードなどが可能となる。

### 4.3. Web ベースでのプロセッサ開発

Web ベースでのプロセッサ開発の流れを示す。

1. プロジェクトの作成
   C-DASH サーバは、プロジェクト用のディレクトリをサーバ上に作り、プロジェクトで必要となるファイルのコピー（プロジェクト管理用の Web ページや、シミュレータを実行するための JNLP など）を行う。
2. プロセッサファイルのアップロード
   開発者は、Web ブラウザを利用し、プロセッサ定義ファイルの生成を行う。アップロードされた定義ファイルは、SRC（Source）ディレクトリに置かれる。
3. CVS リポジトリの作成
   CVS の機能を利用し、サーバ上にリポジトリの登録を行う。
4. 開発者の登録
   プロジェクト管理者により、開発者の登録やアクセス制限の設定を行う。
5. サブプロジェクトの作成
   プロジェクトに登録された開発者は、専用の開発スペースを持つことができ、自由にソースの変更を行うことができる。
6. プロセッサ定義ファイルの編集
   機能の追加や、バグの修正を行う。
7. HW/SWDE の生成
   Verilog-HDL 記述、シミュレータ、アセンブラ、逆アセンブラ、コンパイラの生成を行う。
8. 変更の登録
   変更した内容は、RES（Reservation）ディレクトリに登録される。登録した以外の開発者が承認する場では、元のファイルに反映されない。

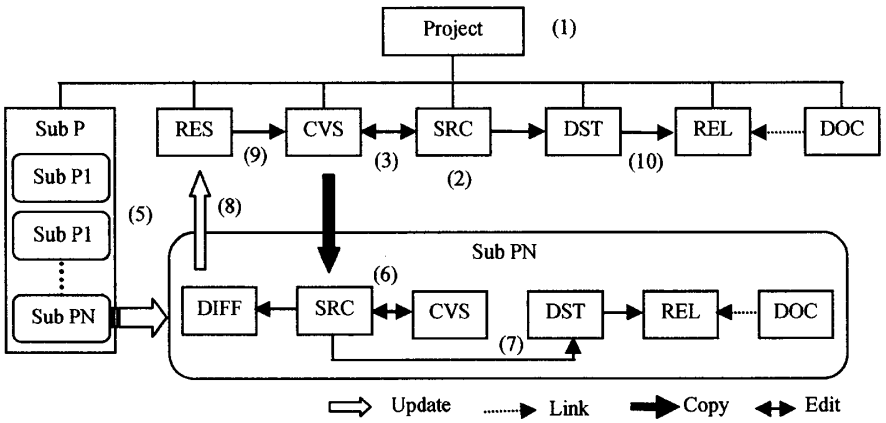


図 2　プロジェクト開発

9. 変更の承認
   登録された内容が正しく実行できかを確認し、正しければ、承認する。承認することで、変更が反映される。
10. リリース
    承認されたファイルを基にし、HW/SWDEの生成を行う。
11. バグレポート
    BBSにバグの報告をする。
12. バグの修正
    BBSに報告されたバグを優先順位に従い修正する。

### 4.4. コミュニケーションツール

C-DASHサーバでは、コミュニケーションツールとして、BBSを提供する。開発者は、自分がどの命令を開発、または、修正するかを宣言してから開発を行う。バグの登録も、BBSを利用して行う。

### 4.5. Webベースのテスト環境

本研究では、ISAベースでのプロセッサ定義を前提としている。このため、開発者は、1命令ごとに、動作定義を追加していくことになる。

このような場合、1命令ごとにIDEの生成、ダウンロード、実行を繰り返すことは、無駄が多くなってしまう。このため、シンプルなWebベースISAシミュレータを提供する。

テストパターンをアップロードすることで、新たに定義した命令が正しく動作しているかを確認することが出来る。

### 4.6. リリースシステム

開発者が、新たに定義した命令やバグの修正が正しく行われているとは限らない。このため、C-DASHサーバでは、RESディレクトリを用意している。

新たに定義された命令や、バグの修正は、一度、RESに登録される。登録された変更内容を他の開発者が承認するまで、プロジェクトのリポジトリに反映されることはない。

承認された更新に基づき、プロジェクトのCVSリポジトリに登録する。

また、変更されたファイル更新時の影響をチェックする。それまでの実行テスト結果と変更を加えた後でのテスト結果が異なるのかをチェックする。（変更により、意図しない影響が及ばないように）

ファイルがアップロードされるたびに、自動的に過去のテストパターンを実行し、実行結果が、前回の実行結果と等しいかのチェックを行う。

この機能により、開発者の意図しない変更が他の命令に影響を与えないことを確認することが出来る。

## 5. 評価

C-DASHで今までに開発した、Intel8080、Java、日立SH、MICROCHIP PIC、DLXプロセッサの記述レベル（DL：Description Level）動作レベル（B：Behavior、S：Sequential、P：Pipeline）、定義命令数（NOI：Number Of Instruction）、記述行数（LOC：Line Of Code）、ゲート数（Gate Size）を表1に示す。

C-DASHでは、命令セットに含まれる、同一の動作記述をグループ化（simgroup、asmgroup）することで、冗長な記述を省くことが出来、少ない記述量で開発を進めることが出来る。

表1 C-DASHでのプロセッサ開発

| Processor | DL | NOI | LOC(行) | GS(ゲート) |
|---|---|---|---|---|
| 8080 | B | 71 | 450 | - |
| Java | B | 201 | 900 | - |
| SH | B | 143 | 1000 | - |
| PIC | S | 37 | 300 | 14000 |
| DLX | P | 55 | 400 | 17000 |
| Java | P | 56 | 300 | 13000 |

## 6. まとめ

本論文では、Webベースで共同してプロセッサを開発するための環境について述べた。これまでは、プロセッサを開発する場合、開発者はプロセッサ開発に必要なツールを開発者それぞれのローカル環境に用意しなければならなかった。また、それぞれの開発者が独自に開発を進めてしまうことによる、ファイルの整合性問題など、チームでの効率的な開発を行えない問題があった。

本研究では、プロセッサ開発をチームで共同して行い、効率的な開発を可能とするプロセッサ開発のためのWCDEの構築について述べた。開発環境をWebベースにすることで、開発者は、テキストエディタとWebブラウザさえあれば、世界中のどこからでも開発に参加することが出きる。

## 参考文献

[1] Hideaki Yanagisawa, Minoru Uehara, Hideki Mori, "A Processor Development Environment C-DASH", International Journal of Computer Science and Network Security (IJCSNS), pp.227-233, Vol. 6, No. 1, January 2006.
[2] Hideaki Yanagisawa, Minoru Uehara, Hideki Mori, "Automatic Generation of a Simulation Compiler by a HW/SW Codesign System", In Proc. of 15th IEEE International Workshop on Rapid System Prototyping (RSP'04), pp.53-59,2004.
[3] CHESS/CHECKERS, http://www.retarget.com/
[4] Xtensa, http://www.tensilica.com/
[5] ASIP Meister, http://www.eda-meister.org/asip-meister/
[6] LISATek, http://www.coware.com/
[7] CVS, http://ximbiot.com/
[8] SourceForege.net, http://sourceforge.net/
[9] GNU Compiler Collection (GCC) Internals, http://gcc.gnu.org/onlinedocs/gccint/
[10] Java Web Start, http://java.sun.com/j2se/1.5.0/docs/guide/javaws/index.html