プロジェクト実験 P2 CPU の論理設計 **岡本 吉央**
平成 18 年 12 月 12 日 **3 回目資料**

1 ソフトウエア工学における要求と仕様 (一般論)

1.1 ソフトウエア・プロセス

工学が扱う対象は「プロダクト」(product) と「プロセス」(process) である．プロダクトは最終製品だけではなく，その過程で生成される中間製品や文書なども含めた生産物を指す．一方，プロセスはプロダクトを生み出す工程である．工学の対象としてのソフトウエア開発プロセスが単に「ソフトウエア・プロセス」(software process) と呼ばれることが多い．

- ライフサイクル・モデル (life cycle model)：ソフトウエアの要求分析，設計，実装，テスト，運用，保守，そして最終的な廃棄に至る過程のモデル．ソフトウエア・プロセスのモデルとして最も一般的．今から挙げていくモデルはどれもライフサイクル・モデルである．

  ちなみに,「life cycle」という英語は生物の生活環を表すことばである．生物の生活環は生物個体の一生を表現するだけではなく，個体の生殖によって世代交代が行なわれる面を重視している．しかし，ソフトウエアのライフサイクルでは世代交代を重視するわけではなく，1 つの製品の一生を表現しているだけであることが多い．また発達心理学においてもライフサイクルという用語があり，これは人間の精神発達の過程を 8 つの段階に分けたものである．

- ウォーターフォール型ライフサイクル・モデル (waterfall model)：「要求分析→設計→実装→テスト→運用・保守」という直線的な流れでライフサイクルをモデル化したもの．

- V 字型ライフサイクル・モデル (V-shaped model)：ウォーターフォール型ライフサイクル・モデルにおけるテストの部分をその前までの工程 (要求分析，設計，実装) を分解して，実装に対して単体テスト，設計に対して統合テスト，要求分析に対してシステムテストをそれぞれ対応させられるようにしたもの．ウォーターフォールに比べてテスト・プロセスをより意識したモデルになっている．

- 反復型ライフサイクル・モデル (iterative model)：要求分析から実装までの開発プロセスをただ一度だけ実行して運用システムを構築するのではなく，はじめは小さな機能範囲のシステムを実現し，その改良を繰り返すことで柔軟にシステムの完成度を上げていくモデル．

  これを突き進めたものの 1 つがエクストリーム・プログラミングやアジャイル開発である．(2 回目資料の「テスト駆動開発」を参照のこと．)

1.2 要求分析と仕様

- 要求 (requirement)：構築すべきシステムの利用者，発注者，またその他関係者がそのシステムに備わっていると考える性質．システムの関係者 (ステークホルダー (stakeholder) と呼ばれる) の持っている要求が明示的なものであるか潜在的なものであるか分からない点がソフトウエア開発を難しくしている一因である．
  - 機能要求 (functional requirements)：システムの振る舞いや内部構造に関する要求．
  - 非機能要求 (nonfunctional requirements)：機能要求ではない要求．多肢に渡り，例えば，性能 (performance)，可用性 (availability)，セキュリティ (security)，保守性 (maintainability)，価格 (price)，安定性 (stability)，信頼性 (reliability) といったもの．

- 要求分析 (requirements analysis)：ステークホルダーの持つ要求を引き出して，構築すべきシステムを定義する作業．要求分析は大きく分けて 2 つの部分から成る．

  1. 問題分析：ステークホルダーの要求を引き出す部分．また，そのために「誰がステークホルダーなのか」ということも分析する必要がある．
  2. 仕様記述：問題分析を基にして構築すべきシステムの定義を記述する部分．

- 仕様 (specification)：要求分析によって得られたシステム定義．仕様記述モデルまたは仕様記述言語によって記述される (1 回目の「モデル化」を参照のこと)．要求と同様に仕様にも様々な分類を考えることができる．

2　ソフトウエア工学に関する参考書

個人的な嗜好でいくつか挙げる．私 (岡本) 自身がソフトウエア工学そのものに関する書籍をそれ程読まないので，かなり偏りがあることは了承していただきたい．

- 玉井哲雄,「ソフトウェア工学の基礎」，岩波書店，2004 年.
  (ソフトウェア工学に関して，モデル化技法を中心とした系統的な解説を格調高く行なっている．)

- B.W. Kernighan，R. Pike,「プログラミング作法」，福崎俊博訳，ASCII，2000 年.
  (ソフトウエアの設計と検証に関して，実例を豊富に用いて解説している．原著は 1999 年出版．)

- ジョン・ベントリー,「珠玉のプログラミング」，小林健一郎訳，ピアソンエデュケーション，2000 年.
  (プログラミングの実践に関わる著者の経験がいきいきと書かれている古典的なベストセラー．ちなみに，私 (岡本) はアルゴリズム理論のある国際会議でこの本の著者の講演を聞いたが，さすがに講演も上手だった．非常に感銘を受けた．)

- S. McConnell,「コードコンプリート (上)(下)」，(株) クイープ訳，日経 BP ソフトプレス，2005 年.
  (ソフトウエア開発に関して示唆に富む実践的技法が紹介されている．厚くて高価だけれどもお薦め．)

- リチャード・ワーマン,「理解の秘密」，松岡正剛訳，NTT 出版，1993 年.
  (経営組織論っぽい本ではあるけれども，ソフトウエア技術者にとってコミュニケーション能力が重要であるという調査結果を鑑みると，ソフトウエア技術者の卵が読んでおくとよい本だと思う．私 (岡本) が大学 3 年生のときに読んで感銘を受け，それ以来いろいろなところで薦めている．今となっては若干記述に古くささを感じるところがあるかもしれない．)

3 CPU の論理設計

3.1 設計する CPU

- 設計する CPU の特徴 (実験指導書 106 ページ)
  - RISC アーキテクチャ：小さな命令セットで高速処理を実現
  - ハーバード・アーキテクチャ：命令用とデータ用にメモリおよびバスを分離
  - ロードストア・アーキテクチャ：演算はレジスタ間で実行．演算対象はメモリからレジスタにロードし，演算結果はレジスタからメモリにストアする．
- 設計する CPU の仕様：配布ファイル p2/cpubeh.vhd に設計仕様が記述されている (すなわち，このファイルはビヘイビア・レベルで CPU を記述している)．
  - 命令セット：実験指導書 108 ページおよび 128〜132 ページを参照．各命令に対するマイクロ動作は 133〜136 ページを参照．
  - 汎用レジスタ：0〜15 までの 16 個．ただし，0 番レジスタは常に値 0 を保持し，レジスタ 15 は JAL 命令の飛び先アドレスを保持するものとする．

3.2 実験手順

1. 配布ファイル p2/cpubeh.vhd に書かれた動作仕様 (および実験指導書) に基づいて，CPU を設計する
2. しかし，配布ファイルでは設計する CPU がある程度できあがっている．本実験では残された部分を穴埋めする形で CPU を完成させる．
3. 配布ファイル p2/readme.???を参照すると,「穴埋め有り」と書かれたファイルと「穴埋め無し」と書かれたファイルがあることに気付く．前者が穴埋めの対象である．
4. 始めに全体像を把握する (トップダウン)
   - p2/CPU.vhd が設計する CPU の本体であるが，ここには各モジュールが配置されるだけである．
     - 8〜21 行目：エンティティ宣言．CPU への入力と出力が定められている．入力の方で，Clk はクロック，Reset はリセット信号，DMdatain はデータメモリからの入力，IMdata はインストラクションメモリからの入力を表す．出力の方はそれぞれ何を表しているのだろうか?
     - 24 行目から最後まで：アーキテクチャ宣言．この部分で各モジュールがどのように結合されているか，読み取ることができる．
     - 26 行目：パッケージの呼出し．このパッケージは配布ファイル p2/CPUCMP.vhd で定義されていて，この中でコンポーネント宣言が行なわれている．(パッケージ設計の詳細についてはサブテキスト第 4 章を参照．) p2/CPU.vhd をコンパイルする前に p2/CPUCMP.vhd がコンパイルされていなくてはならない．
     - 28〜52 行目：内部信号の宣言．
   - p2/cpupac.vhd は CPU 全体，および個々のモジュールで用いられるパッケージを提供している．他のファイルをコンパイルする前に p2/cpupac.vhd がコンパイルされていなくてはならない．
   - [ただちに行なう必要はない．ただし，演習 7 まで到達したら行なうこと．]
     p2/CPU.vhd を見ながら，CPU の各モジュールの接続関係を図示してみる．各モジュールの入力信号と出力信号の対応を考えること．

5. 各モジュールの設計とシミュレーション

(a) 作業ディレクトリは p2 であるので，cd コマンドでそこへ移動する

(b) 全てのファイルで用いる p2/cpupac.vhd を vhdlan コマンドでコンパイルする．

(c) [演習 1] ALU の設計．ALU は p2/alu1.vhd で設計する．ファイルはほとんど完成しているが一部未完成なので，その部分を埋めることで完成する．実験指導書 112 ページの記述と p2/cpubeh.vhd を参照．

- p2/alu1.vhd の 20〜21 行目は変数宣言であり，variable が登場する．信号 (signal) との違いなどについてはサブテキスト第 4 章を参照のこと．
- ファイル中の DMtype，DATA_size，Value1，Value0 などは p2/cpupac.vhd で定義されているので，そちらを参照すること．

(d) [演習 1 続き] ALU の動作確認．動作確認用の VHDL プログラムは自分で一から作成する．ファイル名，コンフィグレーション名などは自分で決めてよい．

- まず，何をすれば動作確認ができるか考え，それからプログラムを作成すること．機能テストの観点からは，Input1，Input2 に可能な全てのテストケースを試すことは膨大すぎて不可能なので，同値分割や限界値分析の考えを適用してみよ．構造テストの観点からは，網羅度が高くなるようにテストケースを選定してみよ．
- シミュレーションの結果，プログラムに誤りが発見されたら，それを修正して再度シミュレーションを行なうこと．
- シミュレーションをちゃんと行なうと (あるいはよく考えると) SLT 演算の結果が想定通り (例えば，実験指導書 129 ページの SLT に書かれている機能仕様通り) ではないことが分かると思う．これは，p2/cpubeh.vhd に書かれた仕様が間違っており，それに基づいて p2/alu1.vhd を作成すると誤りが混入するためである．本実験では，この誤りを修正しなくてもよいが，余裕がある場合はこの誤りも修正すること．

(e) [演習 2] 1 ビットシフト回路の設計．シフト回路は p2/sft.vhd で設計する．

(f) [演習 2 続き] 1 ビットシフト回路の動作確認．ALU と同様に行なう．

(g) [演習 3] レジスタファイルの設計．レジスタファイルは p2/regfu.vhd で設計する．25 行目で宣言している GR_file という内部信号がレジスタの内部状態であり，つまり，これがレジスタに蓄えられる信号である．

(h) [演習 3 続き] レジスタファイルの動作確認．レジスタへの入力にクロックがあるので，それについては中間レポート課題を参照のこと．

(i) [演習 4] 命令デコード回路の設計．ファイルは p2/decode.vhd である．

(j) [演習 4 続き] 命令デコード回路の動作確認．先と同様に．

(k) [演習 5] プログラムカウンタの設計．ファイルは p2/pcu.vhd である．ここからはファイル p2/cpubeh.vhd の中にそっくりそのまま対応する部分があるわけではないので，仕様全体を見て設計する必要がある．実験指導書 133〜136 ページのマイクロ動作表も参照のこと．

(l) [演習 5 続き] プログラムカウンタの動作確認．先と同様に．

(m) [演習 6] メモリ回路の設計．ハーバード・アーキテクチャであるので，命令用のメモリ (命令メモリ IM) とデータ用のメモリ (データメモリ DM) がある．命令メモリのファイルは p2/IMU.vhd で，データメモリのファイルは p2/DMU.vhd である．命令メモリにおいて，命令レジスタ (IR) とメモリアドレスレジスタ (MAR) はプロセスとしてモデル化されていて，データメモリにおいても，メモリアドレスレジスタ (MAR) とメモリバッファレジスタ (MBR) がプロセスとしてモデル化されている．

(n) [演習 6 続き] メモリ回路の動作確認．先と同様に．

(o) [演習 7] シーケンサの設計．ファイルは p2/CNTL.vhd である．全体像を把握する段階で CPU の各モジュールの接続関係を図示していなかったら，この段階で必ず行なうこと．

(p) [演習 7 続き] シーケンサの動作確認．先と同様に．

6. 各モジュールの統合

(a) 穴埋めの必要がなかった p2/ALUU.vhd，p2/B3IU.vhd の内容を確認して，それぞれ vhdlan でコンパイルする．また，p2/CPUCMP.vhd もコンパイルする．

(b) p2/CPU.vhd をコンパイルする．これでモジュールを統合して，CPU 全体が完成した．

7. CPU の動作確認

(a) p2/CPUtest2.vhd が CPU シミュレーションを記述したファイルである．命令メモリとデータメモリを表すテキストファイルを用意して，そこから順にビット列を読み込んで CPU に渡すことをする．

(b) CPU に渡すマシン語プログラムの例も配布ファイルに含まれている．例えば，p2/romIM2 はそのようなプログラムで，それが参照するデータは p2/ramDM2 である．

(c) 実際にこのプログラム例を走らせる前に，p2/romIM2 を逆アセンブルして，どのような動作をすることが期待されるか考えてみること．

(d) シミュレーションを記述したファイル p2/CPUtest2.vhd をコンパイルする前に，49〜50 行目

```
file INPUTDM: TEXT is in "/home/ics-usr/mine/jikken/ramDM2";
file INPUTIM: TEXT is in "/home/ics-usr/mine/jikken/romIM2";
```

の部分を自分のディレクトリにあるファイルが読み込まれるように修正せよ．例えば，

```
file INPUTDM: TEXT is in "/home/ics-06/e123456/p2/ramDM2";
file INPUTIM: TEXT is in "/home/ics-06/e123456/p2/romIM2";
```

とする．

(e) この修正を行なったら，p2/CPUtest2.vhd をコンパイルして，シミュレーションを行なってみる．

8. 自作マシン語プログラムによる CPU の動作確認

(a) 動作確認のためのマシン語プログラムを自分で作成する．実験指導書 115 ページにあるように様々な命令を含めること (これは網羅度を上げるためである)．できる限り意味の分かりやすい「何か」を計算するようなプログラムにするとよい．

(b) 次に，他の学生が作成したマシン語プログラムを自分の作成した CPU で実行させてみる．ちゃんと実行できるだろうか．