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

2003－CE－71 (2)
2003／10／17

Handel-Cによる教育用マイクロプロセッサSEP-3の設計と工数評価

山口 直人[*1]　　塩見 彰睦[*2]

SEP-3は、静岡大学情報科学科の計算機教育用に開発されたマイクロプロセッサである。現在は回路図入力によるボトムアップ式で設計演習を行っている。一方、実際の設計現場では言語ベースのトップダウン式設計手法へのシフトが起きようとしており、本学の設計演習もこのような変化に対応する必要がある。そこで本研究では、C言語を拡張したHDLであるHandel-Cを用いたトップダウン方式のSEP設計方法を提案し、その工数を評価した結果を報告する。実験の結果、目標工数36時間未満を達成することができ、回路規模も現行の実験ボードに実装可能な規模であることが確認できた。

A design and effort evaluation of educational processor "SEP-3" by Handel-C

YAMAGUCHI Naoto　　SHIOMI Akichika

In our university, the design exercise of microprocessor SEP-3 for education using a schematic entry CAD is performed with bottom-up method. On the other hand, design site is try to change to top-down design method using C base hardware description language. We also have to apply to such change. In this paper, we propose the top-down design of SEP-3 using Handel-C, and describe the design effort. At the result, new design method needed 15 man-hour and required approximately 10,000 gates.

1. はじめに

本学では、教育用マイクロプロセッサSEP-2（Shizuoka Educational Processor ver. 2）を用いた教育を行ってきた。本年度より実験用ボードが九州工業大学で設計されたKITEマイクロプロセッサボードPLUS+から、豊田工業高等専門学校で開発された教育用FPGAボードに変更されたため、周辺回路を変更したSEP-3（SEP ver. 3）が使用されている。

現在のカリキュラムは、SEP（SEP-2とSEP-3の両方を指す場合、SEPと表記する）用アセンブラ・逆アセンブラを作成するソフトウェア実験Ⅰ（3年次前期）、SEP用コンパイラを作成するソフトウェア実験Ⅱ（3年次後期）、汎用ロジック回路を使用した順序・組合せ回路の演習を行うハードウェア実験Ⅰ（3年次前期）、CAD（Computer Aided Design）を用いた回路図入力によるSEPの設計演習を行うハードウェア実験Ⅱ（3年次後期）で構成される。

回路図入力によるマイクロプロセッサの設計演習は、論理回路レベルでマイクロプロセッサの動作の理解を目的としている。詳細な動作が理解できるという利点がある一方で、いくつかの問題が顕在化している。

一つ目は、実際の設計現場ではトップダウン方式が主流であるにもかかわらず、ボトムアップ方式で設計を行っていることである。

VLSIの集積率向上にともなってハードウェアが大規模化・複雑化し、それに対応するためにSystemC, SpecCなどに代表されるC言語ベースのより高い抽象度での設計手法が提案されている。実際の設計現場もC言語ベースのトップダウン式設計手法へのシフトが起きようとしており、本学の演習も将来的にはこのような情勢の変化に対応する必要がある。

二つ目は膨大な工数である。SEPアーキテクチャはKITE[1]などのアキュームレータ型アーキテクチャに比べて複雑である。さらに工数のかかる回路図入力を採用しているが、演習時間は午後半日が12回と、Verilog-HDLによるKITE設計演習の演習時間（午後半日が10回）と比べて2回多いだけであるため、演習を時間内に消化できないことがある。

数多く報告されている教育用プロセッサの多くは、設計にハードウェア記述言語(以下HDLと呼ぶ)を用いることで、教育効果と工数を両立している。我々の研究室でもVHDLによるSEP設計演習の検討を行ったが、VHDLの講義を新たに設ける必要があった。

本論文では、別途言語の講義を新たに設けることなく2つの問題を解決するSEP設計方法を提案する。提案設計手法では、近年使用されるようになってき

[*1] 静岡大学大学院情報学研究科
The graduate school of information, Shizuoka university
[*2] 静岡大学情報学部
Fuculty of information, Shizuoka university

たC言語ベースのHDLの一つでありFPGAへの実装に適したHandel-Cを用いてトップダウン設計を行う。

以降、2章では本研究で用いるHanlel-Cと可読性向上のための命名規則について述べる。3章ではSEPアーキテクチャと実験環境について説明する。4章では、SEP-3をHandel-Cで記述する方法とそれに伴う変更点について解説する。5章では評価実験とその結果について述べる。

2. Handel-Cによる設計

2.1 Handel-Cの概要

Handel-Cは、Celoxica社が開発したC言語をハードウェア記述用に拡張した言語である。

記述レベルは、RTLと動作レベルの中間に位置する。Handel-Cの記述は、シミュレーション専用の一部の記述以外は論理合成可能である。

Handel-Cの文法は、遅延のない同期回路設計に抽象化されており、図1 (a)に示すような逐次記述を、変数から変数への代入文1行を1クロックの動作として図1(b)のように合成される。1クロックで複数の動作を並列に行いたい場合は、図2(a)に示すようにpar構文を用いて記述することで、複数行の動作を図2(b)に示すように1サイクルの動作として合成できる。

並列記述の場合、図2中の4行と図2中の5行の動作は同時に行われるため、図2中の5行の右辺にあるcは図2中の3行のc = 0で代入された値0になる。その結果、図1中の5行で変数dに代入される値が3であるのに対し、図2中の5行でdに代入される値は0になる。

signalオブジェクトは、代入にクロックを必要としない変数型である。図3の例では各行が動作するクロックは図3中の(b)に示すように図2の例と同じだが、4行と5の行の動作が異なる。図2の例では、4行の左辺cへの代入は5行のdの演算に反映されずdの値は0になったが、図3の例ではdの値は3になる。

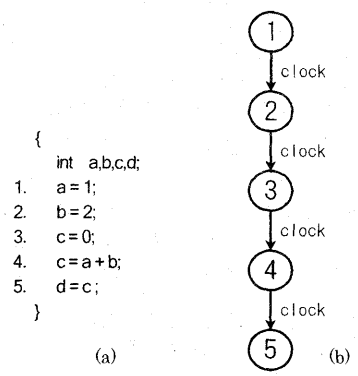

図1. Handel-Cの逐次記述例

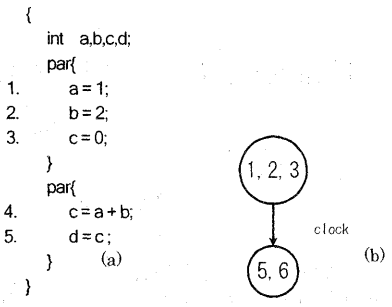

図2. Handel-Cの並列記述例

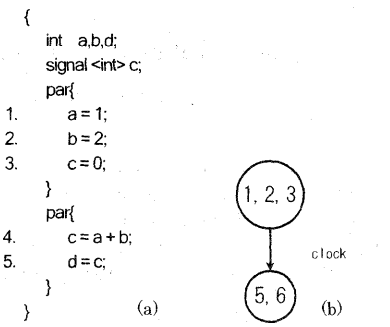

図3. signalオブジェクトを用いた記述例

Handel-Cの変数はレジスタ、signalオブジェクトは配線に近い働きをする。signalオブジェクトへの代入文は、配線の接続ではないため、代入を行ったクロックの間だけ値が保持される違いがある。

本学では、1年次後期よりC言語の演習を行っており、学生は3年次までに一定レベルのC言語のコーディングスキルを身につけている。Handel-Cは、C言語を学習した学生であれば容易に理解できることが報告されており[2]、従来のHDLを用いる場合に比べ、新しく学習する部分が少なく、短時間での習得が期待できる。

2.2 命名規則

C言語の文法を拡張した言語であるため、Handel-C記述の可読性は基本的にはC言語と同等である。しかし、二つの点に注意を払う必要がある。

一つは、図1～図3に示した違いを理解し、クロック単位の動作を把握した上で記述しないと、意図した動作を記述できないことである。

もう一つは、変数のビット幅を任意に指定できるように拡張されており、異なるビット幅や符号の扱いが異なる変数間で演算・代入を行う場合は、演算に必要なビット幅にしてから演算を行い、代入先の

ビット幅に合わせてから代入を行わなければならないことである。これを怠った場合、コンパイル時にエラーになるか、ビット幅が不足して正しい結果を得ることができなくなる。

この二つの点に注意して記述するために、我々はHandel-C 特有の型情報を変数名に反映する命名規則を提案する。我々が提案する命名規則は、Role Field、Data Width Field、Name Field の 3 つのフィールドに分かれており、フィールド間をアンダーバーで区切りことで区別する。Role Field には、その変数の役割を記述し、Data Width Field には、その変数の符号の有無とデータ幅を記述する。Name Field は、変数名を記述するフィールドである。表 1に Role Field の、表 2 に Data Width Field の対応表を示す。Role Field は 1 文字である必要はなく、その信号が複数の意味を持つ場合、優先度が高い順に左側から列挙する。Width Field は符号の有無を示すいずれかの文字に続いて、その変数のデータ幅を 10 進数で記述する。図 4に示す例は、負論理 32bit 符号なしの signal オブジェクトの変数あることを示している。

表 1. Role Field の対応表

意味	prefix文字
signalオブジェクト	s
変数がポート, バスに直結	p
変数が負論理	n

*複数のprefixを用いる場合は表の上から順に書く

表 2. Width Field の対応表

意味	符号付き	符号なし
prefix	n	u

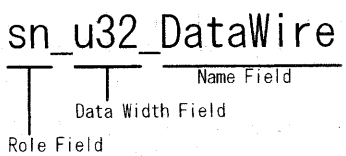

図 4. 提案記法の記述例

本研究の方法として、従来の設計演習で学生が設計を行わない部分はすべてライブラリとして提供することとした。ライブラリへのアクセスはグローバル変数を介して行う。この変数に提案する命名規則を用いることで、利用者が扱いやすいように配慮した。以降の記述例には命名規則を適用する。

3. SEP と実験ボードの概要

3.1 SEP の概要

SEP は、1998 年の SEP-1 に始まり、アーキテクチャの問題点を修正した SEP-2、実験ボード変更にともない周辺回路を変更した SEP-3 が開発されているが、基本的なアーキテクチャの変更は行われていない。SEP アーキテクチャの特徴を表 3に、ブロック図を図 5に示す。

表 3. SEP の仕様

16bitマイクロプロセッサ
- 16bitデータバス
- 16bitアドレスバス
命令セット
- 1語1命令の命令形式(45命令)
- 4種類のアドレッシングモード
- 単一の命令フォーマット
リソース
- 8個の汎用レジスタ(R0～R7)
R5をプロセッサ状態語(PSW)と共用
R6をスタックポインタ(SP)と共用
R7をプログラムカウンタ(PC)と共用
- 4個の専用レジスタ
メモリアドレスレジスタ(MAR)
メモリデータレジスタ(MDR)
命令レジスタ(ISR),バッファレジスタ(B0)
- 16語(SEP2)もしくは81語(SEP3)のI/O空間
- 64k語のメモリ空間

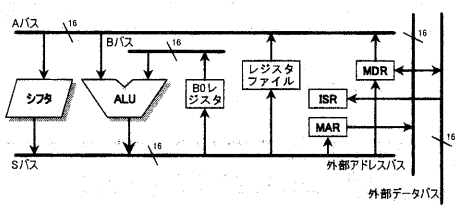

図 5. SEP-3 のブロック図

CPU の命令実行は、図 6に示すように、IF0, IF1, FF0, FF1, FF2, TF0, TF1, EX0, EX1 の 9 状態の遷移で構成され、他に IT0, IT1, IT2 の 3 つの割り込み状態が存在する。

PSW(Processor Status Word:プロセッサ状態語)には N(ネガティブ：演算結果が負のとき 1)、Z(ゼロ：演算結果が 0 のとき 1)、V(オバーフロー：演算結果があふれた時 1)、C(キャリー：桁上げが起きた時 1)フラグが保存される。

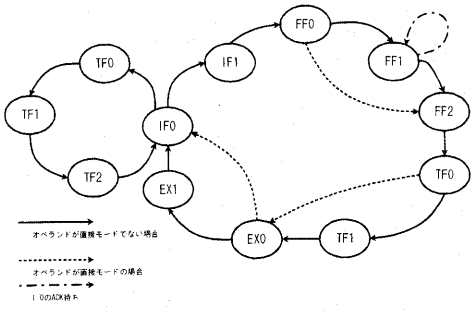

図 6. SEP-3 の状態遷移図

提案する設計方法では、図 6の状態遷移図に基づいて、動作の詳細化を行う。

ALU,シフタの記述は“＋”,“－”のような演算子レベルで記述を行うが、PSW に保存される NZVC フラグは設計者が導出しなければならない。

3.2 SEP の設計演習環境

SEP-3 の実装には、豊田工業高等専門学校で開発された教育用 FPGA ボード（本学では SEP ボードと呼んでいる）を使用する。ボードの仕様を表 4に示す。

周辺入出力のうち、データ入力トグルスイッチ、データ出力 LED は、メモリマップド I/O での利用を考慮して ROM/RAM のアドレスバスと配線を共有している。8×8 ドットマトリクスは、配線の都合で単位時間あたり 1 行 8 ピクセルの制御しかできない。

これらを制御する回路は、従来の SEP-3 設計演習では既設計の部品として与える部分である。提案する設計手法は従来の SEP-3 設計演習と同じ範囲を扱うため、設計対象外とした。設計対象外の回路はライブラリとして別途開発し、学生に提供する。

表 4. 教育用 FPGA ボードの仕様

SEP実装用FPGA
- 米国Altera社製 FLEX 10K30EQC208-3
周辺入出力
- 8セグメントLED(16進表示) × 4
- 8 × 8ドットマトリクス2色LED × 4
- データ入力トグルスイッチ × 16
- データ出力LED × 16
- 設定入力トグル・スイッチ × 4
- 状態出力LED × 4
- ROM/RAMソケット × 8

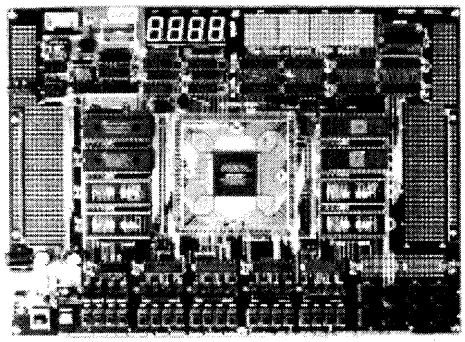

図 7. SEP ボード

周辺入出力のうち、8 セグメント LED、設定入力トグルスイッチ、状態出力 LED、ドットマトリクスは FPGA に接続されている。そのため、接続されているピンを I/O として定義することで、スイッチの状態取得や、LED の点灯/非点灯を行うことができる。

特にドットマトリクスは、行と列の信号線の組合せで点灯/非点灯を行う。そのため、全ての行の点灯処理を行うためには 8 ステップの処理が必要になる。これは、Handel-C の設計フローを学習するのに適当な難易度であるため、演習課題として取り組むことも可能である。

4. Handel-C による SEP の設計

SEP の設計演習は、レジスタ設計に始まり、シフタ・ALU の設計、ステータスカウンタの設計、データ転送制御回路、周辺回路との接続の順にボトムアップ式で行われている。対して、本研究で提案する方法では、大局的な動作から記述するトップダウン式の設計を行う。

設計の方法としては、図 5に示したブロック図をもとに記述するのではなく、図 6の状態遷移図を記述し、各状態の動作を段階的に詳細化してゆく記述方法を採用する。そのため、Handel-C を用いて設計する SEP-3（以後従来の SEP-3 と区別するために SEP-3H と記述）は、SEP-3 と各状態の動作のレベルで互換がある。

SEP-3H の設計で最初に記述を行うのは、SEP の設計演習では 4 番目に行うステータスカウンタである。ステータスカウンタは、SEP の命令実行状態を管理するステートマシンであり、図 6の状態遷移図の動作を実現させるためのユニットである。従来の SEP では、構造が状態遷移図に近い、D フリップフロップを数珠つなぎにしたリングカウンタを用いて実現されている。しかし SEP-3H では、C 言語で状態遷移図をコーディングする手法と同じように、図 8に示すように、現在の状態を示す状態遷移変数と、switch - case 文を

用いて記述する。これは、バイナリカウンタを用いた実現に相当する。

図 8中の(1)行は IF0 状態が無条件に IF1 状態に遷移する記述である。設計初期はこのような記述で、各クロックの動作をシミュレーションで確認する。

汎用レジスタ、MAR (Memory Address Register)、MDR (Memory Data Register)、ISR (Instruction Register) などのレジスタは、変数として定義する。Handel-C の変数はレジスタと同じ働きをするので、Handel-C でレジスタの機能を記述する必要はない。

ALU とシフタの動作は、ステータスカウンタから分離して記述する。これは、Handel-C がコード中の演算子すべてに対してハードウェアを生成するため、冗長な加減算機・ビット演算器を生成しないように、ALU やシフタを利用する演算はデータパスを介して演算するように制御するためである。

```
switch(SEP31Resource->StatusCounter)
{
case MAINSTATE_IF0 :
   SEP31Resource->StatusCounter
       = MAINSTATE_IF1                    } (1)
  break;
case MAINSTATE_IF1 :
       ⋮
case MAINSTATE_IT2 :
  //IT2 での動作
}
```

図 8. SEP-3H のステータスカウンタ記述

```
macro proc ALU(s_u16_ABUS, s_u16_BBUS,
s_u16_ALUSBUS, s_u5_ALUControlCode, s_u4_NZVC)
{
  if(s_u5_ALUControlCode == ALUCC_ADD)   (1)
  par{
    s_u17_ABUSin
      = 0 @ s_u16_ABUS;                  (2)
    s_u17_BBUSin
      = 0 @ s_u16_BBUS;                  (3)
    s_u17_ALUSBUSout
      = s_u17_ABUSin + s_u17_BBUSin;     (4)
    s_n16_ALUSBUS
      = s_u17_ALUSBUSout[15:0];          (5)
    s_u1_Carry
      = s_u17_ALUSBUSout[16:16];         (6)
          ⋮
  }
  else
    ⋮
}
```

図 9. ALU の記述例

ALU の記述は、算術演算子とビット演算子を、シフタの記述は、変数のビットフィールド指定を用いる。これを、図 8のステータスカウンタ記述と並列に動作する処理として記述する。関数呼び出しを用いると、データの受け渡しに変数を用いなければならないため、1 状態を 1 クロックの動作として記述できなくなる。そこで、データの受け渡しに変数ではなく signal オブジェクトが使用できるマクロプロシージャを用いる。ALU のマクロプロシージャ記述例を図 9に示す。

図 9中の(1)行は s_u5_ALUControlCode に格納されている ALU コマンドが“Add” (A+B) である場合の処理を記述することを意味している。図 9中の(2)行、図 9中の(3)行は、加算を行うため 16 ビット符号付き整数を 17 ビットに拡張し、図 9中の(4)行では 17 ビットで加算を行う。これは、Handel-C 記述で図 9中の(6)行のように PSW のキャリーフラグを生成するために必要な処置である。図 9中の(5)行では、17 ビットに拡張したデータ幅を 16 ビットに戻している。

提案する設計手法の ALU は、算術演算子やビット演算子を用いた記述から回路を合成する。そのため、従来の SEP 設計演習で設計していた論理素子単位の動作を利用した技巧的な ALU とは異なったものになるが、この違いは論理回路の講義などで補うことができる。

```
par{
  switch(SEP31Resource->StatusCounter)
  {
       ⋮
  case MAINSTATE_IF1 :
    par{
      s_u16_ABUS = ReadGR(7);                    (1)
      s_u16_BBUS = 0;                            (2)
      s_u5_ALUControlCode = ALUCC_ADD1;          (3)
      s_u16_SBUS = s_u16_ALUSBUS;                (4)
      SEP31Resource->GR[7] = s_n16_SBUS;         (5)
      SEP31Resource->MDR = s_u16_DataReadWire;(6)
          ⋮
    }
    break;
       ⋮
  }

  ALU(s_u16_ABUS,s_u16_BBUS,s_u16_ALUSBUS,
       s_u5_ALUControlCode, s_u4_ALUNZVC);   (7)
  Sifter(s_u16_ABUS,s_u16_SifterSBUS,
       s_u6_SifterControlCode,s_u4_SifterNZVC);(8)
}
```

図 10. SEP-3H コアの記述

図 10は、ステータスカウンタの IF1 ステートの記述である。IF1 ステートでは、命令語の読み込みと、プログラムカウンタのカウントアップを行う。図 10

中の(1)(2)行では ALU、シフタに接続されている signal オブジェクトに演算対象のデータを代入している。ここでは一方にプログラムカウンタの値を、もう一方に 0 を代入している。図 10中の (3)行では ALU の演算モードに“Add and increment” (A+B+1) を指定している。 図 10 中の(4)(5)行では、ALU の演算結果をプログラムカウンタに代入している。図 10 中の(6)行では、ライブラリが提供する外部データバスからの読み込み用 signal オブジェクト s_u16_DataReadWire から MDR にデータを取り込む記述である。

図 10中の(7)(8)行はそれぞれ ALU、シフタをステータスカウンタと並列に動作するようにインスタンス化する記述である。

データ転送制御回路は、Handel-C の代入文から自動的に合成されるので、明示的に記述する必要はない。

周辺回路との接続には、図 10 中の(6)行のように3.2 節で述べたライブラリを使用する。

5. 評価

5.1 実験方法

提案設計手法による工数削減を確認するために評価実験を行った。

実験で工数に計上する範囲は、現在の実験で設計を行っている範囲を設計した時間に限定した。目標工数は、演習 1 回あたり 4.5 時間、全 8 回で設計を終了することとし（全 12 回のうち、Handel-C の学習に 2 回、SEP の復習や成果発表などに 2 回を使用する）、学生一人が全ての設計を行う場合の工数 36 人時未満とした。

設計演習を行う学生は、演習を開始する時点で SEP のアーキテクチャに関する講義を受けている。実験は、回路図入力による SEP 設計演習を行っていないが SEP のアーキテクチャは理解している大学院生が行った。

設計は、全ての記述を一括で行うのではなく、記述とシミュレーションもしくは実機による動作チェックを交互に行い、段階的に詳細化して行った。

5.2 実験結果

SEP-3H を提案設計手法で記述した結果、学生が記述する範囲のコード行数は 1212 行となり、学生が記述可能と思われる行数で記述できることが確認できた。

記述結果を Celoxica 社の DK1.1 Design Suite Service Pack 1 を使用して周辺回路を含めて論理合成を行った結果、必要な論理ゲート数は NAND ゲートに換算すると 10919 ゲートであった。さらに、Altera 社の MAX+plusII 10.2 を使用して SEP ボード上の FPGA に実装した結果、回路を構成するために必要なロジックセルは 1575 個となり、使用率は約 91%であった。従来の回路図入力で設計された SEP-3 は周辺回路を含めて 1044 個のロジックセルを使用し使用率が約 60%であるため、SEP-3H の回路規模は約 1.5 倍になったが、現在の SEP ボードに実装可能な規模である事が確認できた。

SEP-3H コアの工数は、記述・デバッグを併せて 15 人時であった。内訳は記述に 6 人時、デバッグに 9 人時である。この結果、Handel-C による工数削減が確認でき、目標工数を達成できた。

6. おわりに

本研究では、従来の回路図入力によるボトムアップ式の教育用マイクロプロセッサ SEP 設計方法に代わるものとして、教育効果を維持しつつ工数削減を目指す Handel-C を用いたトップダウン式 SEP 設計方法を提案し、その工数を評価した。その結果、工数は 15 人時となり、目標工数 36 人時未満を達成できた。さらに、その回路規模は現在演習で使用している SEP ボード上に実装可能な規模であることが確認できた。

今後の課題として、教育効果に関する評価、実験実施による工数のさらなる検証が挙げられる。

謝辞

本研究は東京大学大規模集積システム設計教育センターを通し、セロクシカ株式会社の協力で行われたものである。

参考文献

[1]. 末吉敏則，久我守弘，柴村英智“KITE マイクロプロセッサによる計算機工学教育支援システム” 電子情報通信学会論文誌，VOL.J84-D-I NO.6，pp917-926，JUNE，2001

[2]. 小松聡，石原享，藤田昌宏“C プログラムからのハードウエア設計教育 －Handel-C と DK1 を使った演習－” DA シンポジウム 2003 論文集，Vol.2003，No.11