

モデル検査に対応する上位ハードウェア 記述言語 Melasy と XML 中間表現

岩崎 直木[†]野村 達雄^{††}和崎 克己[†]信州大学大学院工学系研究科[†]信州大学工学部情報工学科^{††}信州大学大学院工学系研究科[†]

1. まえがき

IT 技術の発展により、様々な環境で電子機器が動作しており、年々規模・数ともに増加傾向にある。また、様々な社会システムがこれらの技術と信頼性に依存しており、とりわけデジタルシステムにおける技術と信頼性は社会システムの維持に必要な不可欠である。仕様通りに確実に動作する信頼性の高いシステムを、より安価に効率よく設計できる環境が望まれている。

ハードウェア設計の正当性を形式的にチェックするツールとしては、SMV, NuSMV 等のモデル検査ツール (Model Checker) [1] が存在する。これらのツールを用いることで設計の正当性の評価を自動で行うことができる。しかし、NuSMV によって実ハードウェアの設計を全て記述するには、極めて低級な言語を利用しなければならない。また、VHDL や Verilog 等の言語で設計したシステムを、検証目的のために、改めて NuSMV 等の言語で記述しなおす工程が新たに発生する。同じ設計を異なる言語で複数回行うことは、実際の開発現場の状況に鑑みるとコスト・納期などの制約において困難である。

本報告は、上記の問題を解決する目的で、一つのコードから、様々な言語処理系に対応するように構築した、上位ハードウェア記述言語 Melasy の設計と、XML による中間表現用のコード生成器の実装について述べたものである [2]。Melasy は C++ [3] [4] のクラスライブラリとして実装されており、Melasy のコードは標準準拠の C++ コンパイラを用いることでコンパイル可能である。また、XML による中間表現を用いることで対象となる言語向けコード生成器の作成を容易にしている。また、モデル検査を行うために、NuSMV 向けコード生成器を作成した。

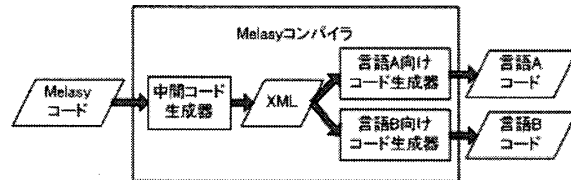


図1 Melasy コンパイラ処理系の位置づけ

2. Melasy の位置づけ

Melasy は NuSMV や VHDL コードなど、様々な処理系向けのコードを生成することを目的とした、上位ハードウェア記述言語である。Melasy と他の言語の関係について、図1に表す。Melasy コンパイラは、中間コード生成器と各対象言語向けコード生成器から構成される。XML による中間表現を平易なものにすることで、各種言語向けコード生成器の作成を容易にしている。

中間コード生成器は、Melasy コードから XML で記述された中間言語を生成するものである。これは C++ のクラスライブラリとして実装されており、melasy.h をインクルードすることで、Melasy のコードを記述できるようになる。ハードウェアを構成するコンポーネントは、C++ のクラスとして記述する。この Melasy コードを C++ コンパイラでコンパイルして得られる実行可能ファイルを駆動することで、XML による中間表現を得ることができる。

3. Melasy の構文上の特徴

Melasy は C++ のコードとして記述するので、文法は C++ の文法に従う。ただし、コンポーネントの動作を記述する箇所では Melasy のライブラリが提供する関数やクラスを継承して記述する必要がある。

Melasy で変数を扱うには、Logic 型と Digit 型の変数を利用する。それぞれ、1 ビットと N ビットの値を表すのに用いる。Logic 型では char 型の '0', '1' のいずれかの値を代入することができる。Digit 型ではテンプレート引数にビット幅を指定することで、任意のビット幅の型として扱うことができ、値の代入には文字列定数か int 型の値を代入することで行う。

ハードウェアを構成するコンポーネントは

A Meta Hardware Description Language Melasy for Model-Checking Systems and its XML Intermediate Representation

[†] Naoki Iwasaki and Katsumi Wasaki, Graduate School of Science and Technology, Shinshu University

^{††} Tatsuo Nomura, Dept. of Information Engineering, Faculty of Engineering, Shinshu University

Component クラスを継承することで定義できる。コンポーネントの動作はコンストラクタ内で定義する。コンポーネントの入出力はコンストラクタ内、in, out, sync 等の関数を呼び出すことで定義できる。出力の定義には C++ の演算子の他 switch_, case_, default_ という特殊な構文を利用できる（コード例[A]を参照）。

[A] コンポーネント定義の例

```
#include "melasy.h"
template<int N>
struct Counter : public Component{
    Logic reset;
    Digit<N> counter;

    Counter(){
        in(reset);
        out(counter);

        sync(counter,
            switch_(reset) [
                case_('1', Digit<N>(0)),
                default_(counter + 1)] );
    }
};

int main(){
    Counter<8> counter;
    std::cout << getXML() << std::endl;
    return 0;
}
```

in や out 関数を複雑な手順で呼び出す必要がある場合も、for や if のような C++ の言語機能をプリプロセッサの様に使うことで、これを簡単に行える。また、再利用可能なコードを関数として定義し、呼び出すこともできる（コード例[B]を参照）。

[B] Logic の配列から Digit を得る関数の例

```
template<size_t N>
Digit<N> toDigit(const Logic(&logics)[N]){
    Digit<N> tmp, swt;
    char one[N+1],
        zero[N+1];
    std::fill(one, one+N, '0');
    std::fill(zero, zero+N, '0');
    zero[N]=one[N]='\0';

    for(int i=0; i<N; i++) {
        one[N-i-1] = '1';
        swt = switch_(logics[i]){
            case_('0', zero), // 0
            case_('1', one)   // 1 << i
        };
        one[N-i-1] = '0';

        tmp = i ? (tmp | swt) : swt;
    }
    return tmp;
}
```

4. XML 中間コード生成

中間コード生成では、インスタンスの存在するコンポーネントのコードが生成される。コー

ド生成は getXML 関数によって行われ、この返り値を標準出力等に出力することで XML 中間コード生成器として動作する。

XML コード生成の特徴 1 つに、生成されたコードから配列が除去される点がある。例えば、array[0] や array[1] といった変数はそれぞれ array_0, array_1 と独立した名前を持つ変数に置き換えられる。これは、各種言語向けコード生成器が配列を扱う必要をなくすため、コード生成器の作成を容易にしている（コード例[C]を参照）。

[C] 生成された中間コード

```
<?xml version="1.0" encoding="ascii"?>
<!DOCTYPE melasy SYSTEM 'melasy.dtd'>

<melasy>
  <component name='Counter_8'>
    <in name='reset' type='Logic' />
    <out name='counter' type='Digit_8' sync='sync'>
      <switch>
        <condition>
          <port type='in' name='reset' />
        </condition>
        <case>
          <const type='Logic' value='1' />
          <const type='Digit_8' value='00000000' />
        </case>
        <default>
          <op name='plus'>
            <port type='self' name='counter' />
            <const type='Digit_8' value='00000001' />
          </op>
        </default>
      </switch>
    </out>
  </component>
</melasy>
```

5. まとめ

上位ハードウェア記述言語 Melasy を導入することで、設計の検証と実装を 1 つのコードで行うことができるようになり、開発のコストを削減することが可能になった。また、C++ のテンプレートなどの強力な機能を使ってハードウェア記述を行うことができる。XML で記述された中間言語の仕様は、各種言語向けのコード生成器の作成を容易にするための工夫がなされており、コード生成器の作成は容易であった。今後は、ハードウェア・ソフトウェア協調設計等への適用を試みたい。

参考文献

- [1] E.M. Clarke, O. Grumberg, D. Peled, "Model Checking," Mit Press (2000).
- [2] V. エイホ, R. セシィ, J.D. ウルマン, "コンパイラ原理・技法・ツール 1, 2," サイエンス社 (1990).
- [3] A. Alexandrescu, "Modern C++ Design," ピアソンエデュケーション (2001).
- [4] Boost, <http://www.boost.org/>.