Artgraphics

初心者の為の SystemVerilogマクロの使い方

篠塚一也
アートグラフィックス

Document Revision: 1.0,2024.02.29

www.artgraphics.co.jp

はじめに

- SystemVerilogには便利で強力なマクロ機能がありますが、使用法を工夫しなければ、実践で役に立つ機能を実現できません。
- このチュートリアルでは、SystemVerilogマクロの使い方を実践的な例を基にして解説します。
- 但し、このチュートリアルではマクロの活用法を重点的に解説する事を目的としているため、ここで試作する機能は限定的です。読者自身で機能拡張を行って下さい。
- 以下で定義するマクロ名やクラス名には gcl_ を付けて名称の矛盾を避けていますが、特別な意味はありません。また、マクロ名は、常に、 _m で終了しています。

目標とするマクロ機能

- このチュートリアルでは、マクロを定義して、以下のようにクラスのメンバーをプリントできるようにします。
- 約90行のSystemVerilogでマクロを実装できます。

```
class sample_t extends gcl_thing_t;
logic [3:0]          a, b, sum;
logic                co;
`gcl_field_begin_m(sample_t)
          `gcl_field_integral_m(a,0)
          `gcl_field_integral_m(b,0)
          `gcl_field_integral_m(co,0)
          `gcl_field_integral_m(sum,0)
`gcl_field_end_m
function new(string name);
          super.new(name);
endfunction
function void add();
          {co,sum} = a+b;
endfunction
endclass
```

```
==========================
Name   Type     Size Value
--------------------------
sample sample_t -    @1
  a    integral 4    'ha
  b    integral 4    'hc
  co   integral 1    'h1
  sum  integral 4    'h6
==========================
```


gcl_print_info_s

- プリントすべき項目をまとめるために便利なストラクチャを以下のように定義しておきます。プリントする項目に対応して定義されているので、内容は分かり易いと思います。
- チュートリアルでは、m_optionを使用しませんが、機能を拡張する時には必要になります。

```
typedef struct {
          string m_tabs, m_name, m_type, m_size, m_value; int m_option;
          }  gcl_print_info_s;
```


gcl_name_m

- 簡単なマクロですが、非常に便利です。このマクロの活用法を身に付けて下さい。

```
`define   gcl_name_m(VAR)     `"VAR`"
```


gcl_field_begin_m

- マクロを開始するマクロを以下のように定義します。
- このマクロを引用すると、print_fields() というファンクションのヘッダが準備されます。
- プリントする項目のリスト pif[$] を定義している事に注意して下さい。

```
`define   gcl_field_begin_m(CLASS_TYPE)                          ¥
          function void print_fields(string tabs);               ¥
          gcl_print_info_s    pif[$], print_info;                ¥
          pif = {};                                              ¥
          print_info = '{tabs,m_name,`"CLASS_TYPE`","-",         ¥
                              $sformatf("@%0d",m_identifier),0};      ¥
          pif.push_back(print_info);
```


gcl_field_integral_m

- プリントすべき項目を定義するマクロで、以下のように定義します。
- 項目のデータタイプにしたがって定義する必要がありますが、このチュートリアルでは、integral型に限定しています。
- プリントすべき項目の情報をセットしたらプリント用の項目リストに追加します。

```
`define   gcl_field_integral_m(VAR,OPTION)                          ¥
          print_info.m_tabs = {tabs,"  "};                          ¥
          print_info.m_name = `gcl_name_m(VAR);                     ¥
          print_info.m_type = "integral";                           ¥
          print_info.m_size = $sformatf("%0d",$bits(VAR));          ¥
          print_info.m_value = $sformatf("'h%h",VAR);               ¥
          print_info.m_option = OPTION;                             ¥
          pif.push_back(print_info);
```


gcl_field_end_m

- プリント用のマクロの終了を以下のように定義しておきます。
- print_fields() ファンクションの終了を示すと共に、プリントの実処理 print_class() を呼び出します。

```
`define   gcl_field_end_m                ¥
          print_class(pif);              ¥
          endfunction
```


gcl_thing_t

- クラスのメンバーのプリント自動処理をするためには、以下のようにベースクラスを定義しておきます。
- print_fields() は前述のマクロで実装されます。
- プリントするためには、print() を呼ぶだけで良いです。

```
class gcl_thing_t;
static int          m_count;
string    m_name;
int       m_identifier;
extern function new(string name);
extern virtual function void print();
extern virtual function void print_class(gcl_print_info_s q[$]);
virtual function void print_fields(string tabs); endfunction
extern function void max_sizes(input gcl_print_info_s q[$],output int nl,tl,sl,vl);
endclass
```


コンストラクタ

- コンストラクタは以下のようになります。

```
function gcl_thing_t::new(string name);
          m_name = name;
          m_identifier = ++m_count;
endfunction
```


print_class()

- クラスのメンバーをプリントする機能は以下のようになります。

```
function void gcl_thing_t::print_class(gcl_print_info_s q[$]);
gcl_print_info_s    print_info;
int        nl, tl, sl, vl, w;
string     format;
           print_info = '{"","Name","Type","Size","Value",0};
           q.push_front(print_info);
           max_sizes(q,nl,tl,sl,vl);
           w = nl+tl+sl+vl+3;
           $display("%s",{w{"="}});
           foreach(q[i]) begin
                     format = $sformatf("%%s%%-%0ds %%-%0ds %%-%0ds %%-%0ds",
                               nl-q[i].m_tabs.len,tl,sl,vl);
                     $write(format,q[i].m_tabs,q[i].m_name,q[i].m_type,
                               q[i].m_size,q[i].m_value);
                     $display;
                     if( i == 0 )
                               $display("%s",{w{"-"}});
           end
           $display("%s",{w{"="}});
endfunction
```


max_sizes()

- プリントする項目群で使用する文字列の幅を計算する目的を持ちます。

```
function void gcl_thing_t::max_sizes(input gcl_print_info_s q[$],
          output int nl,tl,sl,vl);
          nl = 0;
          tl = 0;
          sl = 0;
          vl = 0;
          foreach(q[i]) begin
                    if( nl < q[i].m_name.len+q[i].m_tabs.len )
                              nl = q[i].m_name.len+q[i].m_tabs.len;
                    if( tl < q[i].m_type.len )
                              tl = q[i].m_type.len;
                    if( sl < q[i].m_size.len )
                              sl = q[i].m_size.len;
                    if( vl < q[i].m_value.len )
                              vl = q[i].m_value.len;
          end
endfunction
```


print()

- ユーザが使用するプリント機能の定義は以下のようになります。

```
function void gcl_thing_t::print();
          print_fields("");
endfunction
```


適用例

- 以上で定義した機能を適用するために、クラスを以下のように定義します。

```
class sample_t extends gcl_thing_t;
logic [3:0]          a, b, sum;
logic                co;
`gcl_field_begin_m(sample_t)
          `gcl_field_integral_m(a,0)
          `gcl_field_integral_m(b,0)
          `gcl_field_integral_m(co,0)
          `gcl_field_integral_m(sum,0)
`gcl_field_end_m
function new(string name);
          super.new(name);
endfunction
function void add();
          {co,sum} = a+b;
endfunction
endclass
```


適用例（続）

- テストベンチを以下のように定義します。実行結果は以下のようになります。

```
module test;
sample_t  sample;

initial begin
          sample = new("sample");
          sample.a = 10;
          sample.b = 12;
          sample.add();
          sample.print();
end

endmodule
```

```
==========================
Name   Type      Size Value
--------------------------
sample sample_t -    @1
  a    integral 4    'ha
  b    integral 4    'hc
  co   integral 1    'h1
  sum  integral 4    'h6
==========================
```


まとめ

- クラスのメンバーをプリントする機能を自動生成するSystemVerilogマクロを紹介しました。
- 紹介した機能は限定的ですが、拡張し易い構造になっているので、読者自身で機能拡張を試みて下さい。
- 拡張して欲しい機能は、string型や他のデータタイプのメンバー、アレイ変数、enum変数、オブジェクトのハンドル等です。SystemVerilogの良い練習問題になるので、お試し下さい。

参考文献

- ここで紹介したマクロの機能とプログラミング技術に関しては以下の文献を復習すれば理解できます。

[1] 篠塚一也、SystemVerilog入門, 共立出版 2020.

[2] 篠塚一也、検証のためのSystemVerilogプログラミング、森北出版 2022.

[3] 篠塚一也、SystemVerilog超入門, 共立出版 2023.

- おすすめの書籍