Обзор языка программирования MyHDL

Петров А.В., 28.11.2010

Введение

Язык MyHDL – это язык описания аппаратуры, созданный Джаном Декалювье (Jan Decaluwe) в 2003 году. Язык основан на языке Python, распространяется под лицензией LGPL. Строго говоря, MyHDL – это не полноценный язык, а DSL (Domain Specific Language), реализованный в виде модуля на языке Python. Благодаря возможностям метапрограммирования (прежде всего концепции декораторов) и объектно-ориентированного, функционального программирования в Python библиотека MyHDL расширяет синтаксис и концепции Python для описания оборудования.

Возможности MyHDL позволяют полноценно моделировать и тестировать созданные описания оборудования, генерировать исходные коды на языках Verilog и VHDL – в том числе на их синтезируемых подмножествах.

Судя по истории репозитория MyHDL, инструмент активно развивается и по сей день (последний commit был месяц назад).

MyHDL используется в ряде проектов (список проектов), в том числе и в новых проектах - например, на нем создал проект myBlaze **синтезируемого процессорного ядра MicroBlaze** – автора Jian Luo – проект зарегистрирован на сайте http://www.OpenCores.org в ноябре 2010 г. и доступен в исходных кодах.

Тестовые запуски примеров MyHDL были также проведены автором данной заметки и результаты можно признать вполне успешными.

Преимущества MyHDL по сравнению с Verilog\VHDL отражены авторами на этой странице. К основным преимуществам они относят:

- высокий уровень абстракции языка Python, позволяющий лаконичнее описывать сложные алгоритмы и модели устройств;
- возможность простого моделирования, включая совместное моделирование кода на MyHDL и на Veriog\VHDL (Co-simulation)
- отсутствие сложностей с приведением типов в VHDL, различием блокирующих и не блокирующих присваиваний в Verilog, возможность простой работы с числами со знаком
- возможность одновременной генерации кода как на Verilog, так и на VHDL

К выявленным в ходе изучения недостаткам можно отнести:

1. отсутствие четкого разделения синтезируемого и не синтезируемого подмножеств языка (аналогичная проблема существует и в Verilog, но в MyHDL к ней добавилась проблема наличия еще и третьего подмножества – множества конструкций MyHDL, которые нельзя сгенерировать на Verilog\VHDL)
2. генерируемый Verilog\VHDL код является «плоским» - т.е. при генерации полностью исчезает иерархия проекта – все сигналы объявляются в головном модуле («*The converted output is non-hierarchical because conversion works on a design instance elaborated by the Python interpreter. This process flattens out all hierarchy.*» FAQ_MyHDL)/

К основным концепциям MyHDL можно отнести:

1. Генераторы. Генераторы это понятие языка Python, которое представляет собой реализацию сопроцедур (coroutine) в Python. MyHDL использует генераторы для создания always и комбинаторных блоков, в терминах Verilog – т.е. для описания параллельно функционирующих блоков аппаратуры. При этом введены специальные

декораторы: always, always_comb, instance – позволяющие из любой python-функции сделать генератор соответствующего типа.

2. Сигналы. В MyHDL описан специальный класс Signal – аналог Wire в Verilog
3. Тип intbv – основной тип данных, предназначенный для представления целых знаковых и без знаковых чисел в MyHDL, заданной разрядности, с поддержкой специальных побитовых slice-операций (операций для работы с определенной группой бит)

Подробное описание правил генераций Verilog-исходников по MyHDL программам приведено в полной документации MyHDL в главе 7 (стр. 55), примеры генерации в главе 8 (стр.67)

Примеры на языке MyHDL

Пример 1 - симуляция «Hello world» - файл hello1.py

```
from myhdl import Signal, delay, always, now, Simulation

def HelloWorld():

    interval = delay(10)

    @always(interval)
    def sayHello():
        print "%s Hello World!" % now()

    return sayHello


inst = HelloWorld()
sim = Simulation(inst)
sim.run(30)
```

Результат запуска файла:

```
% python hello1.py
10 Hello World!
20 Hello World!
30 Hello World!
_SuspendSimulation: Simulated 30 timesteps
```

Пример 2 – «Код Грея» - файл bin2gray.py

```
from myhdl import *

def bin2gray(B, G, width):

    """ Gray encoder.

    B -- input intbv signal, binary encoded
    G -- output intbv signal, gray encoded
    width -- bit width

    """
    @always_comb
    def logic():
        Bext = intbv(0)[width+1:]
        Bext[:] = B
        for i in range(width):
```

```
        G.next[i] = Bext[i+1] ^ Bext[i]

    return logic


def main():
    width = 8
    B = Signal(intbv(0)[width:])
    G = Signal(intbv(0)[width:])

    toVerilog(bin2gray, B, G, width)
    toVHDL(bin2gray, B, G, width)

if __name__ == '__main__':
    main()
```

Результат генерации на Verilog

```
// File: bin2gray.v
// Generated by MyHDL 0.6
// Date: Sun Nov 23 11:34:35 2008

`timescale 1ns/10ps

module bin2gray (
    B,
    G
);

input [7:0] B;
output [7:0] G;
reg [7:0] G;


always @(B) begin: BIN2GRAY_LOGIC
    integer i;
    reg [9-1:0] Bext;
    Bext = 9'h0;
    Bext = B;
    for (i=0; i<8; i=i+1) begin
        G[i] <= (Bext[(i + 1)] ^ Bext[i]);
    end
end

endmodule
```

Литература

1. Официальный сайт MyHDL - http://www.myhdl.org
2. Официальный репозиторий - http://sourceforge.net/projects/myhdl/
3. MyHDL в примерах - http://www.myhdl.org/doku.php/cookbook:intro
4. Полная документация по MyHDL - http://www.myhdl.org/doku.php/doc:pdf