# 1 工具需求概述

- 根据yml格式的时钟生成配置描述文件来生成verilog格式的clock generating模块的代码；
- 根据yml格式的时钟生成配置描述文件来生成树状形式的时钟衍生结构示意图。

# 2 yml描述文件说明

时钟发生器模块的结构与配置是通过YAML语言进行描述的。这个YAML格式的时钟发生器描述文件可以分成四个部分，这四个部分分别是：*Top*，*Ports*，*Clock_List*，*Custom_Code*。下面就分别详细说明这四个部分的定义格式及意义。

## 2.1 *Top* 段落

Top段落用来提供对clock generator模块的全局信息的描述。这些全局信息包括了模块的名称、模块的负责人邮件地址、所属项目的名称以及关于这个模块的摘要性描述文字。

Top段落的具体格式如下所述：

```
Top:
  - module: module_name
  - owner: designer@mail.xxxx.com
  - project: project_name
  - Description: >
    Some descripting text about this module.
```

- "**module**: *module_name*" 描述了目标模块的模块名称。
- “**owner**: *designer@mail.xxxx.com*” 描述了目标模块的设计负责人的邮件地址。
- “**project**: *project_name*” 描述了目标模块所属项目的项目代号。
- “**Description**: *some_description_text*” 提供了一些关于目标模块功能和设计目标等的一些简短描述性文字。

下面就用一个具体的例子来说明Top段落的信息应当怎样映射到具体的模块RTL代码。

一个示例Yaml Top段落的描述信息如下所示：

```
Top:
  - module: cmu_clk_gen
  - owner: zhaobeihua@canaan-creative.com
  - project: MAIX-2
  - description: >
    This is the clock signals generating module for the MAIX2 SoC.
    This module generates all the clocks required by the function
    units, modules in MAIX-2 SoC.
```

这段Yaml Top段落描述的信息所对应的Verilog RTL代码如下所示：

```
//------------------------------------------------------------------------------
// Project    : MAIX-2
// Owner      : zhaobeihua@canaan-creative.com
// File Name  : cmu_clk_gen.v
// Module Name: cmu_clk_gen
//------------------------------------------------------------------------------
// Description:
//       This is the clock signals generating module for the MAIX2 SoC.  This
//   module generates all the clocks required by the function units, modules in
//   MAIX-2 SoC.
//------------------------------------------------------------------------------
`timescale  1ns / 1ps
```

## 2.2 *Ports* 段落

Ports段落用来定义由模块设计者显示声明的输入端口信号、输出端口信号以及模块内部的节点信号。这些信号是无法由clkgen生成工具根据具体的clock cell的特性而推断出来，因此必须由模块设计者直接而明确的定义。

Ports段落是一个包含多个signal项的有序列表。而每一个signal项又是一个包含多个 ***map*** 的嵌套 ***map***。 Ports段落的具体定义格式如下所示：

```
Ports:
  - signal_name:
    comment: short description about current signal
    mode: { direction: dir_val, type: type_val, width: NUM }
  - signal_name:
    comment: short description about current signal
    mode: { direction: dir_val, type: type_val, width: NUM }
  - signal_name:
    comment: short description about current signal
    mode: { direction: dir_val, type: type_val, width: NUM }
                        . . . . . .
```

下面就具体的介绍每一个信号项的定义格式和意义。每一个信号项的定义格式如下所示：

```
- signal_name:
    comment: short description about current signal
    mode: { direction: dir_val, type: type_val, width: NUM }
```

- ***signal_name***: 用来定义端口信号（或者模块内部结点信号）的信号名称；
- ***comment***: 用来给出该信号功能或者作用的简短描述；
- ***mode***: 用来对该信号的特征给出更具体的描述（例如，IO方向，信号线类型、信号的比特宽度）。

这里需要对 ***mode*** 项给出更详细的说明。***mode*** 项包含了3个 **map** 类型数据结构项，这三个 **map** 数据项分别提供了关于当前信号的IO方向、信号线类型和信号比特宽度的定义。

1. **direction**: *dir_val*
   **direction** 用来定义当前信号的IO方向，*dir_val* 的合法值可以是：*output*、*input* 和 *node*。*output* 表示当前信号是一个输出端口信号；*input* 表示当前信号是一个输入端口信号；而 *node* 表示当前信号是一个模块的内部节点信号。
2. **type**: *type_val*
   **type** 用来定义当前信号的信号线类型，*type_val* 的合法值可以是：*wire* 或者 *reg*。*wire* 表明当前信号线是由组合逻辑或者模块内的cell instance所驱动的；而 *reg* 表明当前信号线是由时序逻辑（也就是触发器）所驱动的。**请注意！type的默认值是wire，如果在mode定义中没有出现type项，那么当前信号的类型就是wire！**
3. **width**: *NUM*
   **width** 用来定义当前信号的比特宽度。*NUM* 在这里必须是一个大于0的正整数。**请注意！width的默认值是1，如果在mode定义中没有出现width项，那么当前信号的宽度就是1！**

下面就用一个具体的例子来说明Ports段落的信息应当怎样映射到具体的模块RTL代码。

一个示例Yaml Ports段落的描述信息如下所示：

```
Ports:
  - cmu_rst_n:
    comment: This is the global reset to CMU, LOW active
    mode: { direction: input, type: wire, width: 1 }
  - pll_0_clk:
    comment: This is the clock generated from system PLL 0
    mode: { direction: input, type: wire, width: 1 }
  - pll_1_clk:
    comment: This is the clock generated from system PLL 1
    mode: { direction: input, type: wire }
  - pll_2_clk:
    comment: This is the clock generated from system PLL 2
    mode: { direction: input, type: wire }
  - clk_26m:
    comment: This is the 26Mhz clock from off-chip oscilator
    mode: { direction: input }
  - soc_sleep_flag_i:
    comment: the flag signal that the whole SoC entering sleep mode
    mode: { direction: input }
  - sys_bus_aclk_en:
    comment: the clkin enable control for sys_bus_aclk dividor Clk Cell
    mode: { direction: node }
```

这段Yaml Top段落描述的信息所对应的Verilog RTL代码如下所示：

```
module cmu_clk_gen
(
    // in/out port signals defined by module owner
    input wire    cmu_rst_n, // This is the global reset to CMU, LOW active
    input wire    pll_0_clk, // This is the clock generated from system PLL 0
    input wire    pll_1_clk, // This is the clock generated from system PLL 1
    input wire    pll_2_clk, // This is the clock generated from system PLL 2
    input wire    clk_26m, // This is the 26Mhz clock from off-chip oscilator
    input wire    soc_sleep_flag_i, // the flag signal that SoC entering sleep mode
            . . . . . .
);

    //============================================================================
    // declaration of internal wires and registers
    //============================================================================
    wire   sys_bus_aclk_en; // the clkin enable for sys_bus_aclk dividor Cell
            . . . . . .

endmodule
```

## 2.3 *Clock_List* 段落

Clock_List段落是整个描述文件的核心内容之所在。Clock_List段落定义了 *clkgen* 模块所需要生成的所有时钟信号，利用什么类型的clock cell来产生这些时钟信号。同时也隐含定义了表达整个时钟网络中时钟对象派生关系的是时钟树结构图。

Clock_List段落是由多个clock object define项构成的。Clock_List段落的具体定义格式如下所示：

```
Clock_List:
  clk_obj_name:
    comment: short description about clk_obj_name's function or purpose
    mode: { direction: dir_val }
    Source: [ src_clk_obj_0, src_clk_obj_1, ... ]
    Clk_Cell:
      - clk_cell_0_name:
        Param: { PARA0: NUM, PARA1: NUM, PARA2: NUM, ... }
        Pins: { pin0_name: signal_name, pin1_name: signal_name, ... }
      - clk_cell_1_name:
        Param: { PARA0: NUM, PARA1: NUM, PARA2: NUM, ... }
        Pins: { pin0_name: signal_name, pin1_name: signal_name, ... }

  clk_obj_name:
    . . . . . .

  clk_obj_name:
    . . . . . .
```

下面再具体的介绍每一个时钟对象项的定义格式和意义。

1. **clk_obj_name**: 定义了当前时钟对象的名称。
2. **comment**: 用来提供关于当前时钟对象的简短注释性描述；
3. **Source**: 一个父时钟对象的列表。当前时钟对象是通过clock cell根据父时钟对象而派生出来的。父时钟对象列表可以包含一个或者多个父时钟对象；
4. **Clk_Cell**: 用来定义时钟发生器cell列表。时钟发生器是一类 *verilog* module，时钟发生器根据父时钟对象来生成一个新的派生时钟对象。**Clk_Cell** 下可以包含多个时钟发生器cells；
5. **clk_cell_name**: 用来定义具体的一个时钟发生器cell。目前在构建时钟发生器网络时使用到了下列几种时钟发生器cells：
   - `clk2_swi`：两时钟源切换器，从两个时钟源中选择一个作为新的输出时钟；
   - `clk3_swi`：三时钟源切换器，从三个时钟源中选择一个作为新的输出时钟；
   - `clk4_swi`：四时钟源切换器，从四个时钟源中选择一个作为新的输出时钟；
   - `clk_div`：基于 *edge* 的时钟分频器，利用一个源时钟经分频产生一个新的输出时钟；
   - `gate_div`：基于 *clock gate* 的时钟分频器，利用一个源时钟并通过控制clock gate信号而产生一个新的输出时钟；
   - `clk_gate`：clock gating控制cell；
   - `baud_div`：参照UART波特率发生器的工作原理而对源时钟进行分频而产生新的输出时钟；
   - `assign`：verilog中的 **assign** 语句，可直接将输入源时钟重命名后输出。
6. **Param**: *Param* 是 **clk_cell_name** 下的一个属性项。每一种时钟发生器cell都有数个配置参数，不同的配置参数可以使时钟发生器具有不同的特性，从而适应于多种场景的时钟生成。而 *Param* 就是用来定义时钟发生器的配置参数的；
7. **Pins**: *Pins* 是 **clk_cell_name** 下的另一个属性项。