帧格式序列检测生成模块设计规范

Contents

帧格式序列检测生成模块设计规范 1

设计概述 2

子模块概述 3

input_stage 3

async_fifo 6

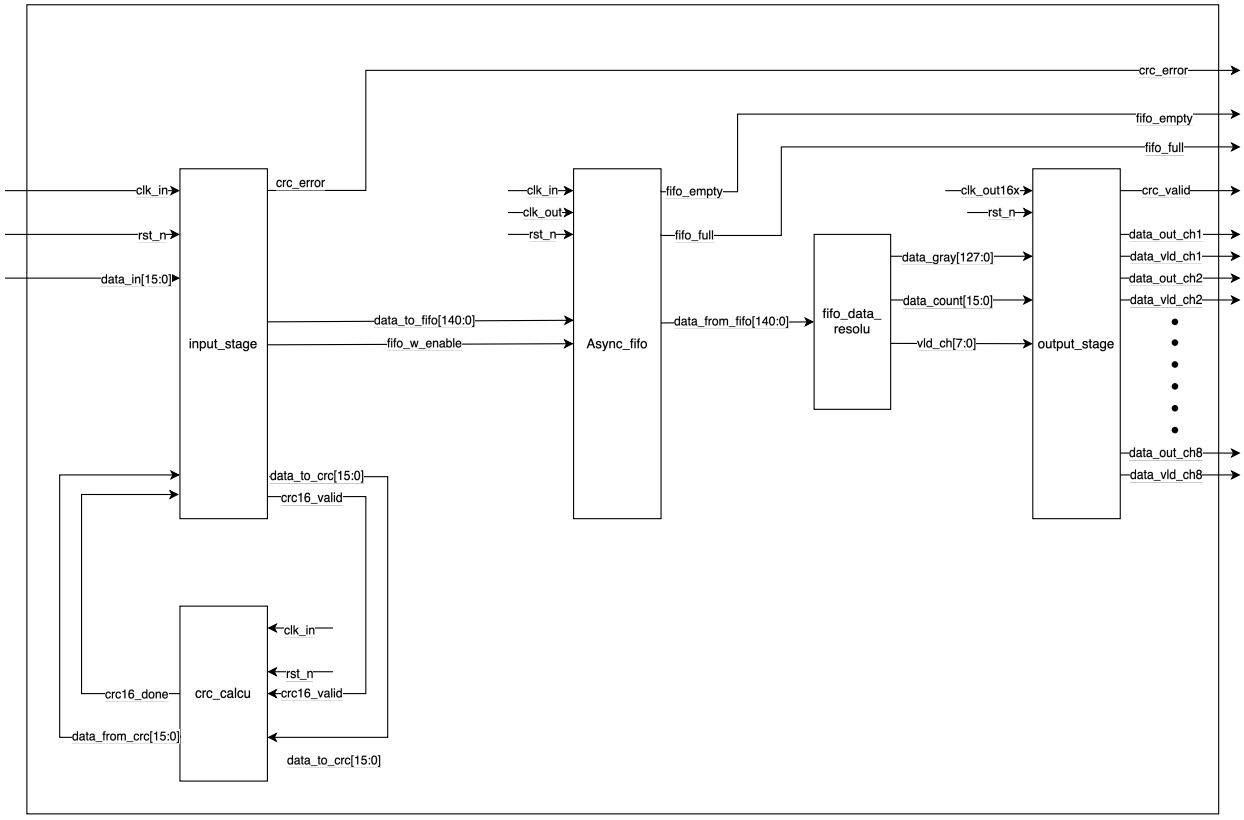
fifo_data_resolu 8

output_stage 9

crc_calcu 10

设计概述

本规范文件描述了帧格式序列检测与生成模块（Frame Format Sequence Generator）的设计规范和特性要求，该模块旨在检测输入数据流中的特定帧格式，并在检测到目标帧时进行串行输出。



如上框图所示，模块接收到数据输入 `data_in` 后，在模块进行以下处理过程：

1. 输入数据检测: 模块接收并采样输入数据。
2. 帧格式识别: 检测并识别输入数据中的特定帧格式，包括帧头、通道选择字段、数据、CRC 校验字段和帧尾 5 个部分。
3. 解帧与数据提取: 识别帧头、通道选择和帧尾，提取数据部分。
4. CRC 校验: 对提取的数据进行 CRC 校验，确保数据完整性。
   - 如果 CRC 校验成功，数据将被写入异步 FIFO 进行缓存。
   - 如果 CRC 校验失败，模块报告错误并丢弃错误数据。

5. 异步 FIFO 缓存: 提取的数据通过异步 FIFO 进行缓存，提供 FIFO 空、FIFO 满信号状态指示。
6. 数据编码: 从 FIFO 中读取数据，按照格雷码进行编码。
7. 根据通道选择进行串行输出: 编码后的数据根据通道选择的数值，决定在哪个通道进行串行输出

子模块概述

input_stage

整体说明：该模块旨在检测输入数据流中的特定帧格式，并提取有效数据。每帧输入 16 位数据，输出 5 种状态的指示信号。具体帧格式如下:

- 帧头: 32 位，取值为 E0E0E0E0
- 通道选择: 8 位, 独热码，高位数据丢弃
- 数据: N 位（可变长度，限制在 16 位到 128 位之间，是 16 的整数倍，按照 Big-Endian 方式输入）
- CRC 校验字段: 16 位
- 帧尾: 32 位，取值为 0E0E0E0E

状态机状态定义:

1.IDLE（空闲状态）

- 初始状态，等待帧头起始
- 行为：持续监测输入数据是否为帧头的第一部分 E0E0
- 转换条件：当 data_in == 16’hE0E0 时进入 HEAD_CHECK

2.HEAD_CHECK（帧头校验状态）

- 验证完整的 32 位帧头
- 行为：检查第二个 16 位数据是否为 E0E0
- 转换条件：若 data_in == 16’hE0E0 进入 CHANNEL 否则返回 IDLE

3.CHANNEL（通道选择状态）

- 读取通道选择字段
- 行为：存储输入数据的低 8 位（高 8 位丢弃）到 data_ch 寄存器
- 转换条件：无条件进入 DATA 状态（仅需 1 周期）

4.DATA（数据接收状态）

- 接收数据字段和后续字段
- 行为：

  启动 4 位计数器（data_counter）从 0 开始计数

  每个周期将 32 位寄存器 tail_detec_reg（用于帧尾检测，初始化全 0）左移 16 位，并存入输入数据 data_in

  每个周期将 160 位寄存器 full_data_reg（用于存储数据字段，初始化全 0）左移 16 位，并存入输入数据 data_in

  计数器递增，溢出时丢弃数据
- 转换条件:当检测到 data_in 为 0E0E0E0E 时进入 CRC_OUTPUT,停止寄存器 tail_detec_reg 和 full_data_reg 的存储若计数器溢出（≥15）返回 IDLE

5.CRC_OUTPUT（CRC 提取状态）

- 提取 CRC 字段并计算数据长度

- 行为:

  从 32 位寄存器 `tail_detec_reg` 中获取 CRC 字段（帧尾前 1 周期的数据）

  计算数据长度：data_count = data_counter - 2

  截取 full_data_reg 的高 128 位作为 data_128

- 转换条件:

  当检测到 data_in 为 0E0E0 时进入 ENABLE_CRC（确保帧尾正确）否则数据错误，返回 IDLE

`6.ENABLE_CRC（CRC 校验使能状态）`

- 使能 CRC 校验模块
- 行为：拉高 start_crc 信号，启动 CRC 校验模块，结果由 CRC 校验模块返回
- 转换条件：**无条件**返回 IDLE

单独为与 CRC 交互编写一个 always 块，确保后续数据的输入在 CRC 校验期间不会被丢弃:

- 行为:

  接收到 start_crc 信号后，拉低 start_crc 信号，crc_cnt 开始计数。

  当 crc_cnt 小于 data_count 的值的时候，维持 crc16_valid 高电平，同时每个周期从 data_128 低位开始每 16 位作为 data_to_crc 发送给 CRC 校验模块。所有数据已发送后清零 crc16_valid 和 crc_cnt。

  接收到 crc16_done 信号后，检查 CRC 结果 data_from_crc 和 crc_field_reg 是否相等。相等时候拉高 fifo_w_enable，拉低 crc_err，data_to_fifo 赋值为 {data_128, data_ch, data_count} 写入 fifo。不相等时拉低 fifo_w_enable，拉高 crc_err。

顶层 IO

信号	位宽	I/O
clk_in	1	I
rst_n	1	I
data_in	16	I
crc_err	1	O
data_to_fifo	140	O
fifo_w_enable	1	O
data_to_crc	16	O
data_from_crc	16	I
crc16_done	1	I
crc16_valid	1	O

信号说明

- `clk_in`: 输入时钟
- `rst_n`: 异步复位信号，低电平有效
- `data_in`: 16 位并行输入数据
- `crc_err`: CRC 校验结果，错误时为真
- `data_to_fifo`: 输出到 fifo 的数据
- `fifo_w_enable`: fifo 模块的写使能信号
- `data_to_crc`: 输入到 CRC 校验模块的待校验数据
- `data_from_crc`: 从 CRC 校验模块返回的数据校验值
- `crc16_done`: 从 CRC 校验模块返回的数据校验值有效信号，高电平有效
- `crc16_valid`: 发送给 CRC 校验模块的使能信号，高电平有效

async_fifo

整体说明: 传统的异步 fifo，宽度为 140 位，深度为 8，提供 fifo 空、fifo 满信号状态指示

顶层 IO

信号	位宽	I/O
clk_in	1	I
clk_out	1	I
rst_n	1	I
fifo_w_enable	1	I
fifo_r_enable	1	I
data_to_fifo	140	I
data_from_fifo	140	O
fifo_empty	1	O
fifo_full	1	O

信号说明

`clk_in`：fifo 输入时钟域的时钟

`clk_out`: fifo 输出时钟域的时钟，与 `clk_in` 为同频异步。

`rst_n`: 异步复位信号，低电平有效

`fifo_w_enable`：fifo 写使能信号

`fifo_r_enable`：fifo 读使能信号

`data_to_fifo`：fifo 输入数据

`data_from_fifo`：fifo 输出数据，当 `fifo_empty` 为 1 时应读出 140‘b0

`fifo_empty`：fifo 空信号

`fifo_full`：fifo 满信号

时序说明

数据将在 `fifo_w_enable` 被拉高的时钟周期内被写入

数据将在 fifo_r_enable 被拉高的时钟周期内被读出

fifo_data_resolu

整体说明：纯组合逻辑模块，将从 fifo 读到的 140 位数据分离为 128 位数据位 +8 位通道选择位 +4 位数据长度表示位。其中高 128 位位数据位，中间 8 位位通道选择位，低 4 位为数据长度表示位。

4 位数据长度表示位生成 16 位具体数据长度位 data_count，例如 0-8 分别输出 0 16 32 48 64 80 96 112 128。

数据位高位在前，长度由 data_count 给出，不足 128 位的数据将在低位补 0。数据位转为格雷码后，在低位补 0 补齐 128 位，输出信号 data_gray 也是高位在前。

8 位通道选择位不做处理，直接输出 8 位信号 vld_ch。

顶层 IO

信号	位宽	I/O
data_from_fifo	140	I
data_gray	128	O
vld_ch	8	O
data_count	16	O

信号说明

data_from_fifo：fifo 输出数据 data_gray：输出数据的格雷码表示 vld_ch：通道选择数据 data_count：具体数据长度位

output_stage

整体说明: 本模块根据前面模块提供的信息，将数据发送到对应的通道并转为串行信号输出。根据 vld_ch 所输入的独热码，将输入的 data_gray 发送到对应的数据串行输出通道 (data_out_ch1~8)，同时拉高对应通道的数据有效信号 (data_vld_ch1~8)。根据 data_count，具体决定输出 data_vld_ch 的持续周期数，即 data_vld_ch 仅在输出时拉高。

顶层 IO

信号	位宽	I/O
rst_n	1	I
clk_out16x	1	I
data_gray	128	I
vld_ch	8	I
data_count	16	I
crc_valid	1	O
data_out_ch1	1	O
data_out_ch2	1	O
...	1	O
data_out_ch8	1	O
data_vld_ch1	1	O
data_vld_ch2	1	O
...	1	O
data_vld_ch8	1	O

信号说明

- rst_n: 异步复位信号，低电平有效
- clk_out16x: 输出串行信号速率的时钟
- data_gray: 输出数据的格雷码表示
- vld_ch: 通道选择数据，8 位宽独热码
- data_count:data_gray 的具体数据长度
- crc_valid: 任一 data_vld_ch* 拉高时即拉高
- data_out_ch1~8: 数据串行输出通道，输出时高位在前
- data_vld_ch1~8: 数据有效信号通道，仅在输出时拉高

crc_calcu

设计一个 CRC 校验模块，采用 CRC-16/CCITT 算法，多项式：0x1021，初始值：为输入的 crc 信号，输入不反转，输出不反转，输出异或值：0x0000。

顶层 IO

信号	位宽	I/O
data_to_crc	16	I
crc	16	I
data_from_crc	16	O

信号说明

`data_to_crc`: 从帧格式识别模块输入的待校验数据

`crc`: 从帧格式识别模块输入的 crc 校验初始值

`data_from_crc`: 输出数据的 CRC 校验值