wgl 串行文件转 pat 文件

(1). 脚本实现功能描述：

从原始 wgl 串行文件的 pattern 模块如图 1 所示（如果原文件中这个模块不是以 pattern 关键词开头，请把其它关键词修改成 pattern）提取信号端口与对应的信号值，然后根据信号功能描述文件，将不是输入或者不是输出的信号端口及对应信号值删除，最后根据 pattern 模块中的不同 test 如图 2，3 所示(第一个 test 数据向量必须以关键词{ chain_test }开头，以关键词{ End chain test }结束，注意空格。第二个 test 以关键词{ scan_test }开头，以关键词 end 结束。如果原文件格式不符，请将原文件关键词进行修改)生成不同的 pat 文件。（脚本默认原文件 pattern 模块中只有两个 test，生成两个 pat 文件）

```
pattern Chain_Scan_test("p_wake_n":I, "p_uart_txd":I, "p_uart_rxd":I, "p_tst4":I,
    "p_test_mode":I, "p_spi_do":I, "p_spi_di":I, "p_spi_csn":I, "p_spi_clk":I,
    "p_perst_n":I, "p_i2c_sda":I, "p_i2c_scl":I, "p_gpio"[7]:I, "p_gpio"[6]:I,
    "p_gpio"[5]:I, "p_gpio"[4]:I, "p_gpio"[3]:I, "p_gpio"[2]:I, "p_gpio"[1]:I,
    "p_gpio"[0]:I, "p_wake_n":O, "p_uart_txd":O, "p_uart_rxd":O, "p_tst4":O,
    "p_test_mode":O, "p_spi_do":O, "p_spi_di":O, "p_spi_csn":O, "p_spi_clk":O,
    "p_perst_n":O, "p_i2c_sda":O, "p_i2c_scl":O, "p_gpio"[7]:O, "p_gpio"[6]:O,
    "p_gpio"[5]:O, "p_gpio"[4]:O, "p_gpio"[3]:O, "p_gpio"[2]:O, "p_gpio"[1]:O,
    "p_gpio"[0]:O)
```

图 1. 原始 wgl 串行文件的 pattern 模块

```
{ Chain_test }
{ Pattern 0 Vector 1 TesterCycle 1 }
{ PLL Capture cycles: procedure = core_capture }
{ Begin chain test }
{ Vector type: TEST_SETUP }
vector(+, gen_tp1) := [ - 0 0 - 1 - - - - 0 0 0 - 0 - - - 0 0 0 X
 - - X - X X X X - - - X - X X X - - - ];
{ Vector type: TEST_SETUP }
vector(+, gen_tp1) := [ - 0 0 - 1 - - - - 0 0 0 - 0 - - - 0 0 0 X
 - - X - X X X X - - - X - X X X - - - ];
```

```
{ Vector type: SHIFT }
vector(+, gen_tp1) := [ - 0 1 - 1 1 - 0 0 1 1 0 - 0 - - - 1 0 0 X
 - - X - - X - - - - - X - X X X - - - ];
{ Vector type: SHIFT }
vector(+, gen_tp1) := [ - 0 1 - 1 0 - 1 1 1 1 0 - 0 - - - 1 0 0 X
 - - X - - X - - - - - X - X X X - - - ];
{ End chain test }
{ Scan_test }
```

图 2. 第一个 test 格式

```
{ Scan_test }
{ Pattern 0 Vector 107061 TesterCycle 107080 }
{ PLL Capture cycles: procedure = core_capture }
{ Pattern 0 Vector 107061 TesterCycle 107080 }
{ Vector type: LOAD_UNLOAD }
vector(+, gen_tp1) := [ - 1 0 - 1 - - - - 1 1 0 - 0 - - - 0 0 0 X
 - - X - X X X X - - - X - X X X - - - ];
{ Vector type: LOAD_UNLOAD }
vector(+, gen_tp1) := [ - 1 1 - 1 - - - - 1 1 0 - 0 - - - 0 0 0 X
 - - X - X X X X - - - X - X X X - - - ];
```

```
{ Vector type: SHIFT }
vector(+, gen_tp1) := [ - 0 1 - 1 1 - 1 1 1 1 0 - 0 - - - 1 0 0 X
 - - X - - X - - - - - X - X X X - - - ];
{ Vector type: SHIFT }
vector(+, gen_tp1) := [ - 0 1 - 1 0 - 1 1 1 1 0 - 0 - - - 1 0 0 X
 - - X - - X - - - - - X - X X X - - - ];
end
```

图 3. 第二个 test 格式

(2). 打开脚本界面如图 4 所示，在原始 wgl 文件路径一栏点击“浏览”选择需要进行转换的 wgl 串行文件，点击“保存两个文件”在文件存储路径两栏选择输出文件保存路径，最后点击“运行”，如果弹出提示界面如图 5 所示，表示转换完成，点击“确定”自动退出界面。

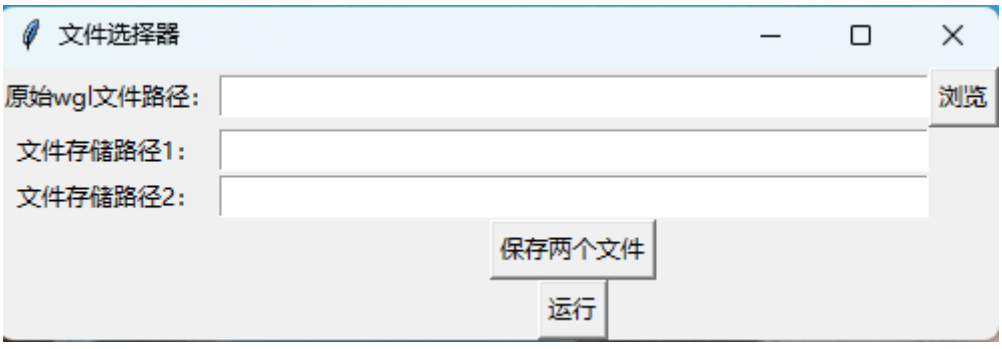

图 4. 脚本界面

图 5. 提示界面

(3). 原始 wgl 串行文件格式要求:

a. 如上述（1）所述如果原文件中这个模块不是以 pattern 关键词开头，请把其它关键词修改成 pattern。

b. 如上述（1）所述原文件 pattern 模块中的 test 格式第一个 test 数据向量必须以关键词{ chain_test }开头，以关键词{ End chain test }结束，注意空格。第二个 test 以关键词{scan_test}开头，以关键词 end 结束。如果原文件格式不符，请将原文件关键词进行修改。

c. 脚本默认原文件 pattern 模块中只有两个 test，生成两个 pat 文件，如果原文件超过两个 test 则脚本不适用。如果原文件只有一个 test，只需将开头关键词改成上述{chain_test}或{scan_test}其中一个，结束改成其对应的关键词即可，这样生成的两个 pat 文件中，只有一个 pat 文件有内容，另一个文件内容为空。

d. 对原文件提取的数据向量内容只包含{ chain_test }\{ End chain test }与{ scan_test }\end 两组 test 之间的数据，如果要提取两组 test 之外的数据，可改变关键词位置，如图 6 所示。

```
{ Pattern 0 Vector 0 TesterCycle 0 }
{ Vector type: PRIMARY }
vector(+, gen_tp1) := [ - 0 0 - 1 - - - - 1 0 0 - 0 - - - 0 0 0 X
 - - X - X X X X - - - X - X X X - - - ];
{ Chain_test }
{ Pattern 0 Vector 1 TesterCycle 1 }
{ PLL Capture cycles: procedure = core_capture }
{ Begin chain test }
{ Vector type: TEST_SETUP }
vector(+, gen_tp1) := [ - 0 0 - 1 - - - - 0 0 0 - 0 - - - 0 0 0 X
 - - X - X X X X - - - X - X X X - - - ];
```

图 6. 修改关键词位置