

Рубежный контроль 1. Описание конечных автоматов на языке Verilog. Вариант №9

Задание. Разработать модуль на Verilog, который реализует конечный автомат, проверяющий соответствие входной последовательности ASCII-символов заданному регулярному выражению. Автомат должен работать на высокой тактовой частоте, а входные данные поступать с UART-совместимой скоростью (входной baudрейт). Выходной сигнал должен выдаваться также с UART-совместимой скоростью (выходной baudрейт). Для детектирования класса ASCII символов необходимо инстанцировать модуль комбинационного устройства `ascii_type_detector` (реализовывать модуль не нужно).

Входной baudрейт: 57600 бод

Тактовая частота: 40 МГц

Выходной baudрейт: 28800 бод

Регулярное выражение для распознавания

`(start_stop)(vowel)(consonant){3}(vowel)(start_stop)`

Описание: Слово из 5 букв (гласная-согласные-гласная)

Пример валидной строки: `\0Apple\0`

Пример невалидной строки: `\0Aplle\0`

Приложение

Основные метасимволы:

`()` - группировка символов
`|` - логическое "ИЛИ" (альтернатива)
`*` - 0 или более повторений предыдущего элемента
`{n}` - ровно n повторений
`{n,m}` - от n до m повторений

Базовые классы:

`start_stop` - нулевой символ (`\0`)
`whitespace` - пробелы, табуляции
`other` - любые символы, не попавшие в другие классы

Буквы и цифры:

`small_letter` - строчные буквы (a-z)
`capital_letter` - заглавные буквы (A-Z)
`number` - цифры (0-9)
`hex_digit` - шестнадцатеричные цифры (0-9, A-F, a-f)
`vowel` - гласные буквы (a, e, i, o, u, A, E, I, O, U)

Символы пунктуации:

`punctuation_basic` - основные знаки препинания (`.`, `,`, `:`, `;`, `!`, `?`, `'`, `"`)
`punctuation_finance` - финансовые символы (`#`, `$`, `%`, `&`, `@`)

Скобки и операторы:

`parentheses` - круглые и квадратные скобки (`()`, `[]`, `<`, `>`)
`curly_braces` - фигурные скобки (`{`, `}`)
`math_symbol` - математические операторы (`+`, `-`, `*`, `/`, `,`, `=`, `<`, `>`)

Примеры интерпретации:

`(number){2,4}` - от 2 до 4 цифр подряд
`(capital_letter | number)` - одна заглавная буква ИЛИ одна цифра
`(whitespace)*` - ноль или более пробелов/табов
`(vowel){2,3}` - последовательность из 2 или 3 гласных букв

Часть 1. Диаграмма переходов состояний автомата (8 баллов)

Регулярное выражение для распознавания:

(start_stop)(vowel)(consonant){3}(vowel)(start_stop)

Часть 2. Описание заголовка модуля, внутренних сигналов и локальных параметров (8 баллов)

Входные данные

clk – тактовый сигнал.

rst – сигнал сброса (активный уровень высокий).

ascii_char[7:0] – текущий принятый ASCII-символ.

char_valid – строб-сигнал, указывающий на валидность *ascii_char* (синхронизирован с UART-приёмом).

Выходные данные

sequence_valid – сигнал валидности всей последовательности (активный уровень высокий).

output_strobe – строб-сигнал, указывающий на актуальность *sequence_valid* (синхронизирован с UART-передачей, но на другом бодрейте).

Часть 3. Инстанцирование модуля `ascii_type_detector` (4 балла)

```
module ascii_type_detector (  
    input wire [7:0] ascii_char,  
    output reg small_letter,          // a-z  
    output reg capital_letter,        // A-Z  
    output reg number,                // 0-9  
    output reg hex_digit,             // 0-9, A-F, a-f  
    output reg punctuation_basic,     // .,:;!?''"  
    output reg punctuation_finance,   // #$$%&@  
    output reg parentheses,          // (), []  
    output reg curly_braces,          // {},  
    output reg math_symbol,           // +-*\/\<>=  
    output reg whitespace,            // пробелы, табы  
    output reg vowel,                 // aeiouAEIOU  
    output reg start_stop,            // \0 (нулевой символ)  
    output reg other                  // всё остальное  
);
```

Часть 4. Описание автомата, два процесса (10 баллов)

Регулярное выражение для распознавания:

(start_stop)(vowel)(consonant){3}(vowel)(start_stop)

**Часть 5. Описание счетчиков-делителей частоты и генерация выходных и внутренних сигналов
(10 баллов)**