МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего профессионального образования

Национальный исследовательский ядерный университет «МИФИ»

Институт Интеллектуальных Кибернетических Систем

Отчет по лабораторной работе

Учебная дисциплина : «Схемотехника»

Тема: Сторожевой таймер

Выполнили:

студенты группы С22-501

Павлюк Глеб

Плотников Артем

Смирнов Илья

Москва 2024

Оглавление

1. Постановка задачи

Реализовать на языке VHDL устройство модифицированного WatchDog таймера (устройство обратного отсчета фиксированного промежутка времени). Выполнить в двух вариантах: поведенческое описание (Behavioral) и набор логических выражений (RTL).

Модификация стандартного Watchdog Timer заключается во внедрении опций остановки, возобновления таймера.

2. Спецификация

2.1. Условное графическое обозначение и список портов ввода-вывода

Таймеру на вход могут поступать следующие сигналы:

`i_clk` — clock.

`i_reset` — сигнал обнуления таймера и начала обратного отсчета.

`i_pause` — сигнал перевода таймера в режим ожидания (обратный отсчёт ставится на паузу).

`i_ticks` — сигнал длины обратного отсчета

`i_update` — сигнал установки длины обратного отсчета

И один выходной сигнал:

`o_time_up` — сигнализирует о конце обратного отсчета.

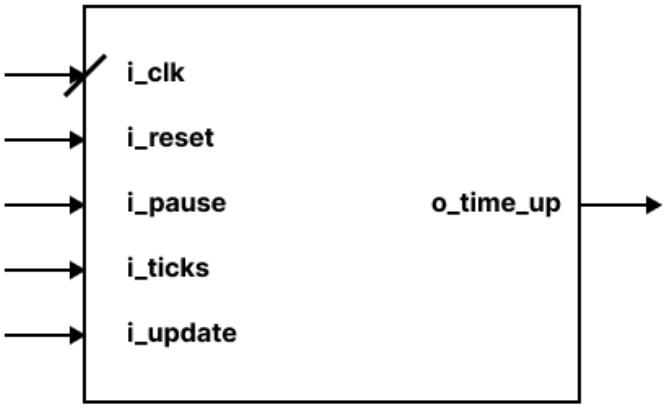

Рис. 1 УГО Сторожевого таймера

Разрабатываемое устройство представляет собой комбинационную логическую схему с 5 входами данных, 1 выходом данных и несколькими управляющими входами, сигналы на которых задают сдвиг между данными на входе и данными на выходе.

2.2. Описание рабочего режима

Разработанный модифицированный сторожевой таймер ведет отсчет установленного времени в тиках, которое представляет собой критическое время зависаний системы. Изначально таймер находится в режиме ожидания `“idle”` (запрет счёта), из которого можно выйти при установке сигнала `i_reset` в значение `‘0’` в состояние отсчета, `“count”`: оно сопровождается выгрузкой количества отситываемого времени из хранящегося значения `tick_limit.` Также предусмотрен функционал постановки таймера на паузу, в состояние `“pause”`.

Систему можно представить в виде конечного автомата. Тогда множество состояний автомата:

- `idle`
- `count`
- `pause`

Рассмотрим его графовое представление.

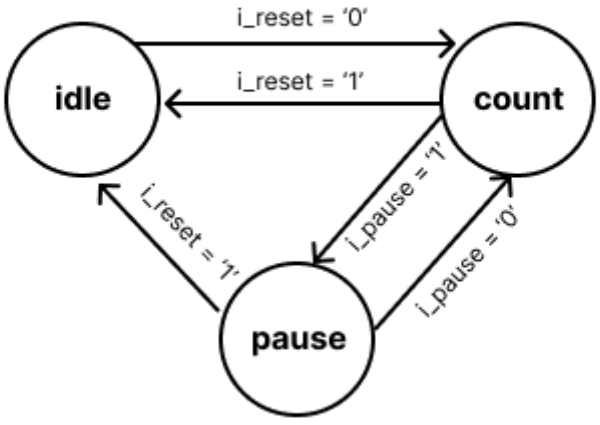

Рис. 2 Представление сторожевого таймера в виде конечного автомата

3. Тестирование

В рамках данной работы требовалось провести тестирование разработанного устройства.

Были сформулированы следующие сценарии тестирования:

1) Задание количества тактов обратного отсчета
2) Нажатие паузы во время отсчета
3) Сброс таймера
4) Окончание времени таймера

Рис. 3 Временная диаграмма - задание количества тактов и пауза

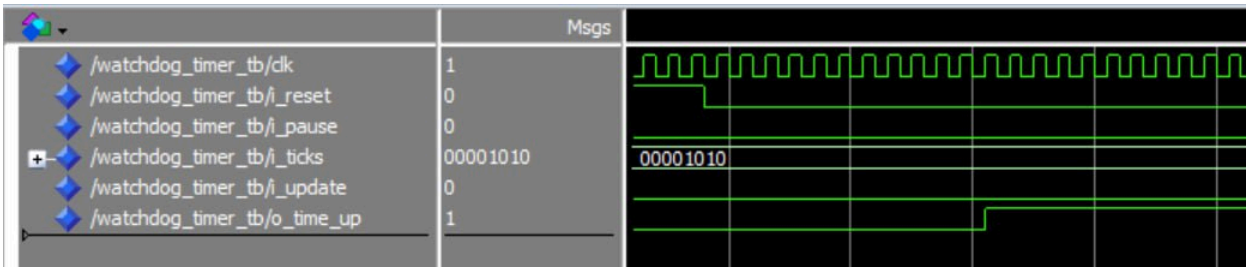

Рис. 4 Временная диаграмма - отсчет и окончание времени таймера

4. Результаты синтеза

RTL схема по результатам синтеза проиллюстрирована на рисунке 4.

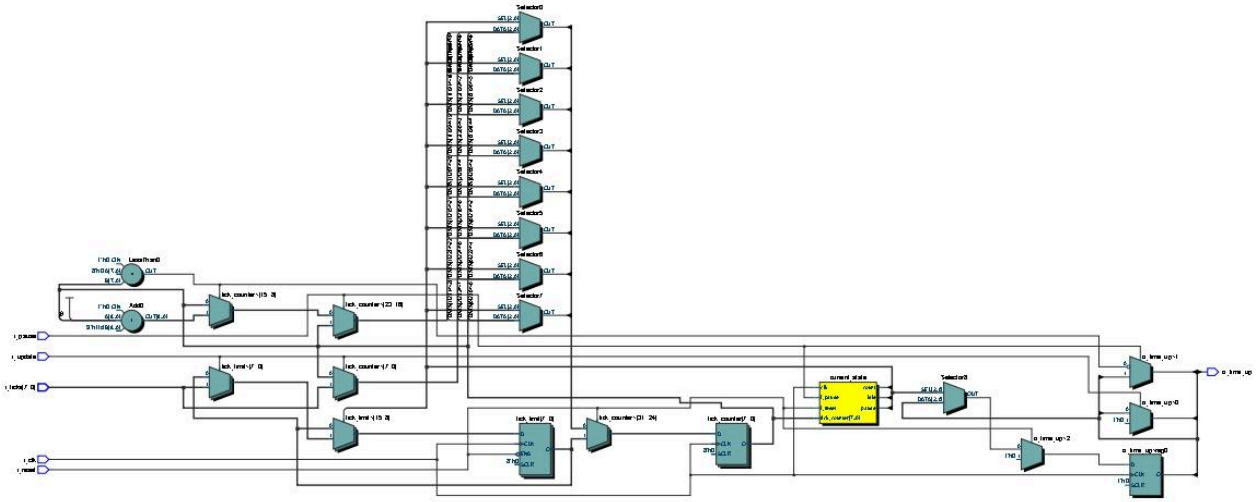

Рис. 4 RTL схема разработанного устройства.

Список затраченных ресурсов среды проиллюстрирован на рисунке 5.

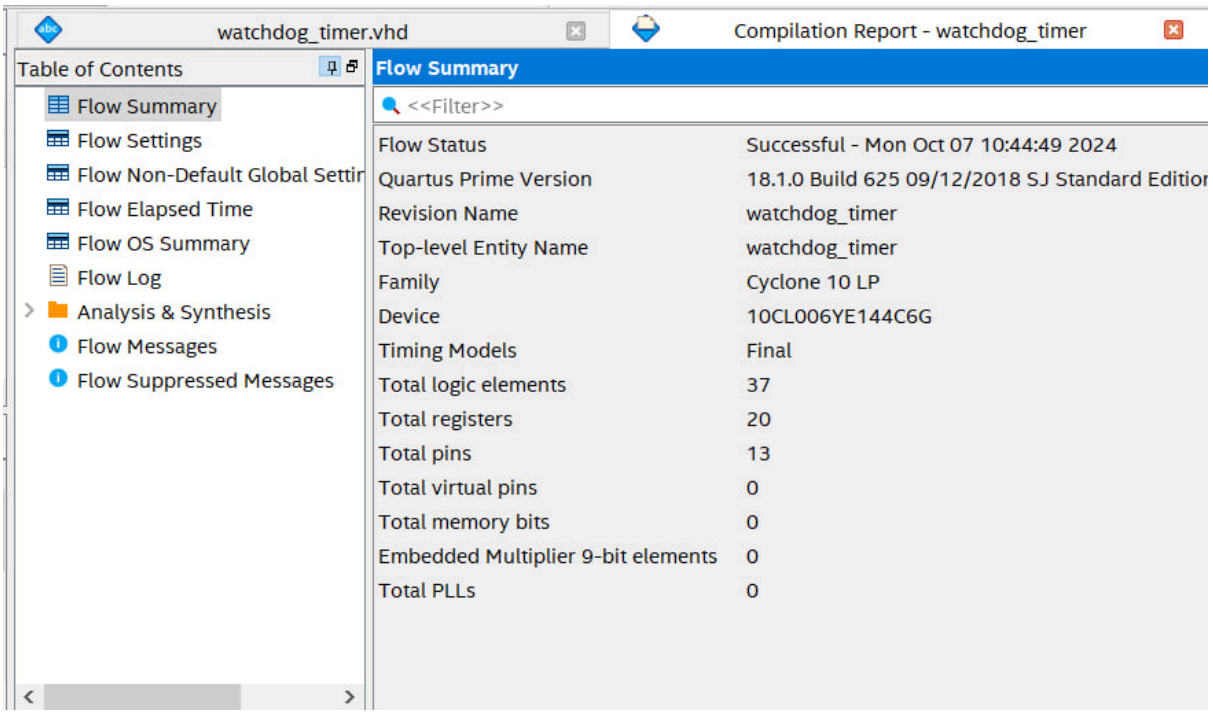

Рис. 5 Статистика затраченных ресурсов.

Приложение

1.1 Файл watchdog_timer.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity watchdog_timer is
    port (
        i_clk     : in std_logic;
        i_reset   : in std_logic;
        i_pause   : in std_logic;
        i_ticks   : in std_logic_vector(7 downto 0);
        i_update  : in std_logic;
        o_time_up : out std_logic
    );
end entity watchdog_timer;

architecture behavioral of watchdog_timer is
    type state_type is (idle, pause, count);
    signal current_state, next_state : state_type := idle;
    signal tick_counter : unsigned(7 downto 0) := (others => '0');
    signal tick_limit   : unsigned(7 downto 0) := (others => '0');
begin

    process (i_clk)
    begin
        if rising_edge(i_clk) then
            if i_reset = '1' then
                current_state <= idle;
                tick_counter  <= tick_limit;
                o_time_up     <= '0';
            else
                current_state <= next_state;
                case current_state is
                    when idle =>
                        if i_update = '1' then
                            tick_limit   <= unsigned(i_ticks);
                            tick_counter  <= unsigned(i_ticks);
                            o_time_up    <= '0';
                        end if;
                    when count =>
                        if i_pause = '1' then

                            null;
                        else
                            if tick_counter > 0 then
                                tick_counter <= tick_counter - 1;
                                o_time_up    <= '0';
                            else
                                o_time_up    <= '1';
                            end if;
                        end if;
                    when pause =>
                        null;
                end case;
            end if;
        end if;
    end process;

    process (current_state, i_pause, i_reset, tick_counter)
    begin
        next_state <= current_state;

        case current_state is
            when idle =>
                if i_reset = '0' then
                    next_state <= count;
                end if;

            when count =>
```

```
                if i_pause = '1' then
                    next_state ⩽ pause;
                elsif tick_counter = 0 then
                   next_state ⩽ idle;
                else
                    next_state ⩽ count;
                end if;

            when pause ⇒
                if i_pause = '0' then
                    next_state ⩽ count;
                else
                    next_state ⩽ pause;
                end if;

            when others ⇒
                next_state ⩽ idle;
        end case;
    end process;

end architecture behavioral;
```

1.2 Файл watchdog_timer_tester.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity watchdog_timer_tester is
    port (
        i_clk     : out std_logic;
        i_reset   : out std_logic;
        i_pause   : out std_logic;
        i_ticks   : out std_logic_vector(7 downto 0);
        i_update  : out std_logic;
        o_time_up : in std_logic
    );
end entity watchdog_timer_tester;

architecture tester of watchdog_timer_tester is
    signal clk : std_logic := '0';
    constant clk_period : time := 10 ns;
begin
    clk_process : process
    begin
        i_clk ⩽ '0';
        wait for clk_period / 2;
        i_clk ⩽ '1';
        wait for clk_period / 2;
    end process;
    test_process : process
    begin
        i_reset ⩽ '1';
        i_pause ⩽ '0';
        i_update ⩽ '0';
        i_ticks ⩽ (others ⇒ '0');
        wait for clk_period * 2;
        i_reset ⩽ '0';

        i_ticks ⩽ std_logic_vector(to_unsigned(10, 8));
        i_update ⩽ '1';
```

```
        wait for clk_period;
        i_update ⩽ '0';

        wait for clk_period * 8;

        i_pause ⩽ '1';
        wait for clk_period * 3;
        i_pause ⩽ '0';

        wait for clk_period * 10;

        i_reset ⩽ '1';
        wait for clk_period * 5;
        i_reset ⩽ '0';

        wait for clk_period * 20;
        wait;
    end process;
end architecture tester;
```

1.3 Файл watchdog_timer_tb.vhd

```
library ieee;
use ieee.std_logic_1164.all;

entity watchdog_timer_tb is
end entity watchdog_timer_tb;

architecture tb of watchdog_timer_tb is
    signal clk     : std_logic;
    signal i_reset   : std_logic;
    signal i_pause   : std_logic;
    signal i_ticks   : std_logic_vector(7 downto 0);
    signal i_update  : std_logic;
    signal o_time_up : std_logic;

    component watchdog_timer
        port (
            i_clk     : in std_logic;
            i_reset   : in std_logic;
            i_pause   : in std_logic;
            i_ticks   : in std_logic_vector(7 downto 0);
            i_update  : in std_logic;
            o_time_up : out std_logic
        );
    end component;

    component watchdog_timer_tester
        port (
            i_clk     : out std_logic;
            i_reset   : out std_logic;
            i_pause   : out std_logic;
            i_ticks   : out std_logic_vector(7 downto 0);
            i_update  : out std_logic;
            o_time_up : in std_logic
        );
    end component;
```

```
begin
    t1: watchdog_timer
        port map (
            i_clk     ⇒ clk,
            i_reset   ⇒ i_reset,
            i_pause   ⇒ i_pause,
            i_ticks   ⇒ i_ticks,
            i_update  ⇒ i_update,
            o_time_up ⇒ o_time_up
        );

    t2: watchdog_timer_tester
        port map (
            i_clk     ⇒ clk,
            i_reset   ⇒ i_reset,
            i_pause   ⇒ i_pause,
            i_ticks   ⇒ i_ticks,
            i_update  ⇒ i_update,
            o_time_up ⇒ o_time_up
        );

end architecture tb;
```