درس: طراحی سیستم‌های دیجیتال fpga / asic

استاد درس: دکتر بالغی

Traffic Light Controller

مریم لیاقت

هدف تمرین: نوشتن کد چراغ راهنمایی در VHDL

توضیح کد:

در ابتدای کد، سیگنال‌های ورودی مانند clk (کلاک اصلی)، stby (حالت آماده‌به‌کار)، و test (حالت تست)، و سیگنال‌های خروجی برای کنترل چراغ‌های راهنمایی تعریف شده‌اند.

همچنین حالت‌های مختلف سیستم (RG, RY, GR, YR, YY) برای نشان دادن وضعیت چراغ‌ها تعیین شده‌اند.

چون کلاک اصلی خیلی سریع است، از یک شمارنده برای تقسیم این کلاک استفاده شده تا یک سیگنال کندتر به نام slow_clk تولید شود.

این سیگنال کند برای کنترل زمان‌بندی چراغ‌ها استفاده می‌شود.

سیستم دارای ۵ حالت است:

RG:مسیر ۱ قرمز، مسیر ۲ سبز

RY:مسیر ۱ قرمز، مسیر ۲ زرد

GR:مسیر ۱ سبز، مسیر ۲ قرمز

YR:مسیر ۱ زرد، مسیر ۲ قرمز

YY:هر دو مسیر زرد

با هر تغییر وضعیت کلاک کند(slow_clk) ، ماشین حالت به وضعیت بعدی می‌رود.

مدت زمان هر حالت توسط متغیر time مشخص می‌شود. اگر حالت test فعال باشد، زمان هر حالت به مقدار کوچکی کاهش می‌یابد (برای تست سریع‌تر).

در هر حالت، خروجی‌های مربوط به چراغ‌ها (r1, r2, y1, y2, g1, g2) به‌صورت مناسب تنظیم می‌شوند تا نشان‌دهنده وضعیت چراغ در آن لحظه باشند.

```
ENTITY tlc IS
    PORT ( clk, stby, test: IN STD_LOGIC;
           r1, r2, y1, y2, g1, g2: OUT STD_LOGIC);
END tlc;

ARCHITECTURE behavior OF tlc IS

    -- Clock Divider Constants
    CONSTANT clk_divider_max : INTEGER := 20000000;  -- Divide 20MHz to 1Hz (adjust as needed)

    -- Timing Constants Based on Divided Clock (1Hz)
    CONSTANT timeMAX   : INTEGER := 54;
    CONSTANT timeRG    : INTEGER := 36;
    CONSTANT timeRY    : INTEGER := 6;
    CONSTANT timeGR    : INTEGER := 54;
    CONSTANT timeYR    : INTEGER := 6;
    CONSTANT timeTEST  : INTEGER := 1;

    -- State Declaration
    TYPE state IS (RG, RY, GR, YR, YY);
    SIGNAL pr_state, nx_state: state;

    -- Timing Signal
    SIGNAL time : INTEGER RANGE 0 TO timeMAX;

    -- Clock Divider Signals
    SIGNAL slow_clk : STD_LOGIC := '0';
    SIGNAL clk_div_count : INTEGER RANGE 0 TO clk_divider_max := 0;

BEGIN

    -- Clock Divider Process (Divide 20MHz to ~1Hz)
    clk_divider_proc: PROCESS (clk)
    BEGIN
        IF rising_edge(clk) THEN
            IF clk_div_count = clk_divider_max / 2 THEN
                slow_clk <= NOT slow_clk;
                clk_div_count <= 0;
            ELSE
                clk_div_count <= clk_div_count + 1;
            END IF;
        END IF;
    END PROCESS;
```

```
    PROCESS (slow_clk, stby)
        VARIABLE count : INTEGER RANGE 0 TO timeMAX := 0;
    BEGIN
        IF stby = '0' THEN
            pr_state <= YY;
            count := 0;
        ELSIF rising_edge(slow_clk) THEN
            count := count + 1;
            IF count = time THEN
                pr_state <= nx_state;
                count := 0;
            END IF;
        END IF;
    END PROCESS;

    -- State Transition Logic
    PROCESS (pr_state, test)
    BEGIN
        CASE pr_state IS
            WHEN RG =>
                r1<='1'; r2<='0'; y1<='0'; y2<='0'; g1<='0'; g2<='1';
                nx_state <= RY;
                IF test='0' THEN time <= timeRG;
                ELSE time <= timeTEST;
                END IF;

            WHEN RY =>
                r1<='1'; r2<='0'; y1<='0'; y2<='1'; g1<='0'; g2<='0';
                nx_state <= GR;
                IF test='0' THEN time <= timeRY;
                ELSE time <= timeTEST;
                END IF;

            WHEN GR =>
                r1<='0'; r2<='1'; y1<='0'; y2<='0'; g1<='1'; g2<='0';
                nx_state <= YR;
                IF test='0' THEN time <= timeGR;
                ELSE time <= timeTEST;
                END IF;

            WHEN YR =>
                r1<='0'; r2<='1'; y1<='1'; y2<='0'; g1<='0'; g2<='0';
                nx_state <= RG;
                IF test='0' THEN time <= timeYR;
                ELSE time <= timeTEST;
                END IF;

            WHEN YY =>
                r1<='0'; r2<='0'; y1<='1'; y2<='1'; g1<='0'; g2<='0';
                nx_state <= RY;
        END CASE;
    END PROCESS;

END behavior;
```

فایل UCF مربوط به کد:

```
NET "clk" LOC =P80;
NET "stby" loc=p101 | pullup;
NET "test" loc=p102 | pullup;
NET "r1" LOC = P108;
NET "y1" LOC = p109;
NET "g1" LOC = p111;
NET "r2" LOC = p115;
NET "y2" LOC = P116;
NET "g2" LOC = P117;
```