MH

Masoud Heidary

think outside the box

CA Lab – EX6

MasoudHeidaryMH@gmail.com

فرمول های استفاده شده

$$\frac{Input\ Clock}{Frequency} = counter \rightarrow InputClock = Counter * Frequency$$

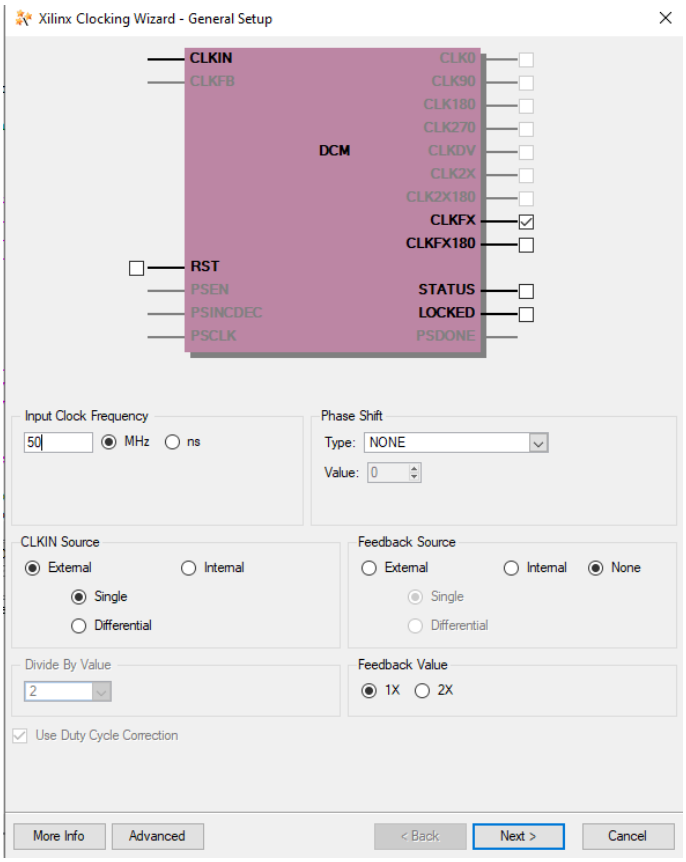
با معلوم بودن فرکانس ورودی و فرکانس مورد نظر، Counter مورد نظر بدست میاید.

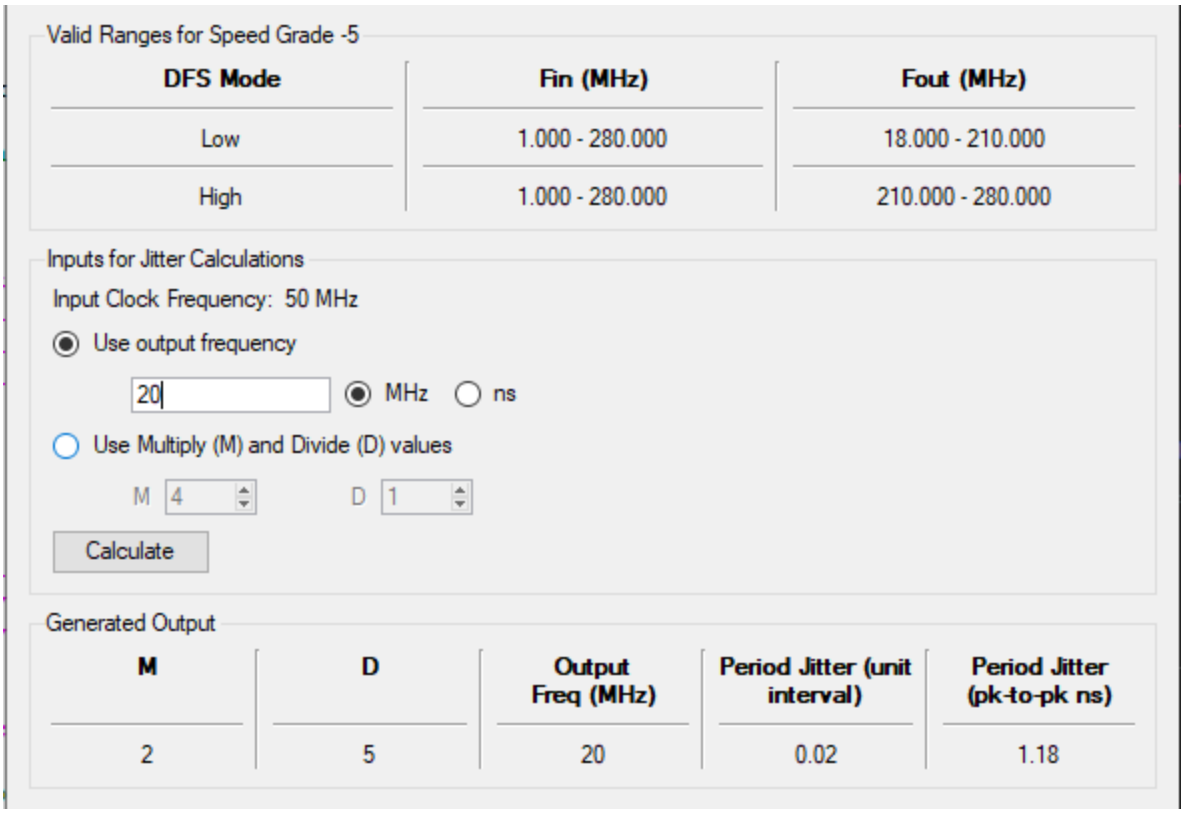
نکته:

- متغیرهای داخل VHDL ما نمیتواند بیشتر از 32 بیت باشد، و این محدودیتی بزرگ است
- اگر فرکانس مرجع ورودی را زیاد کنیم، دقت Duty Cycle ای که میتوانیم ایجاد کنیم بیشتر میشود اما چون شمارنده های ما محدود به 32 بیت میباشد، بازه کلاک خروجی ما محدود تر میشود

طراحی

ابتدا با استفاده از DCM کلاک مرجع را به 20MHz کاهش میدهیم





حال ابتدا برای ایجاد کردن سیگنال خروجی، با Duty Cycle 50% به صورت زیر عمل میکنیم

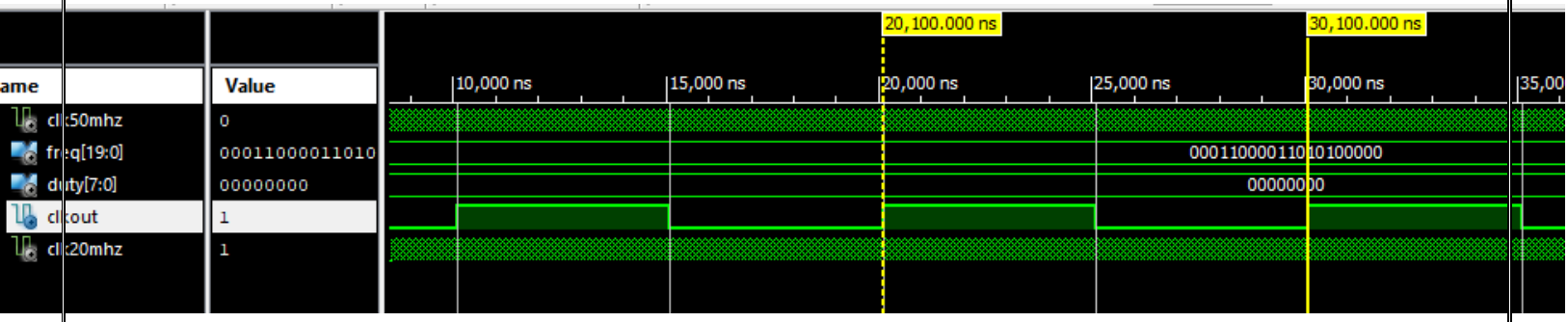
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity ClkDiv is
    Port ( Clk50MHz : in  STD_LOGIC;
           Freq : in  STD_LOGIC_VECTOR (19 DOWNTO 0);
           Duty : in  STD_LOGIC_VECTOR (7 DOWNTO 0);
           ClkOut : out  STD_LOGIC);
end ClkDiv;

architecture Behavioral of ClkDiv is
    COMPONENT ClkDiv_DCM
    PORT(
        CLKIN_IN : IN std_logic;
```

```
        CLKFX_OUT : OUT std_logic;
        CLKIN_IBUFG_OUT : OUT std_logic
        );
    END COMPONENT;

    signal Clk20MHz: STD_LOGIC;
begin

    Inst_ClkDiv_DCM: ClkDiv_DCM PORT MAP(
        CLKIN_IN => Clk50MHz,
        CLKFX_OUT => Clk20MHz,
        CLKIN_IBUFG_OUT => open
    );

process(Clk20MHz)
    variable counter: integer := 0;

    variable clk_state: STD_LOGIC := '0';
begin
    if rising_edge(Clk20MHz) then
        counter := counter + 1;

        -- create clk period
        if counter * to_integer(unsigned(Freq)) = 20_000_000 then
            clk_state := '1';
            counter := 0;
        end if;

        -- create duty cycle
        if counter * to_integer(unsigned(Freq)) = 10_000_000 then
            clk_state := '0';
        end if;

        ClkOut <= clk_state;
    end if;
end process;

end Behavioral;
```

Test Bench:

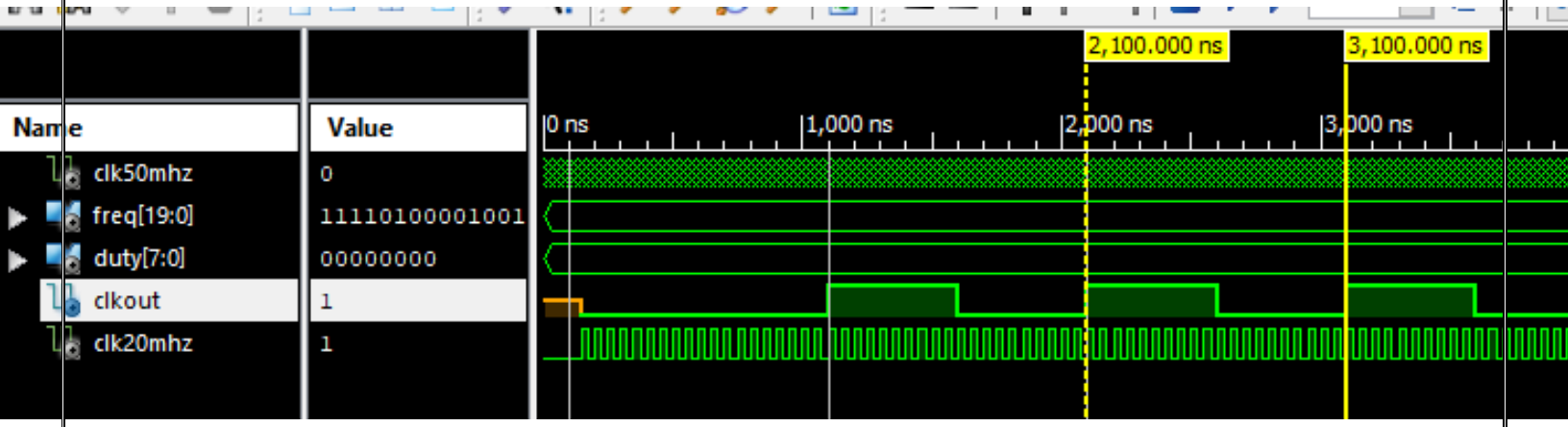
1KHz

```
Freq <= "00011000011010100000";
      wait for 10 ns;
```



1MHz

```
Freq <= "11110100001001000000";
      wait for 10 ns;
```



حال برای اضافه کردن Duty Cycle میتوانیم تکه ای از کد را به صورت زیر تغییر دهیم

From:

```
-- create duty cycle
if counter * to_integer(unsigned(Freq)) = 10_000_000 then
    clk_state := '0';
end if;
```

To:

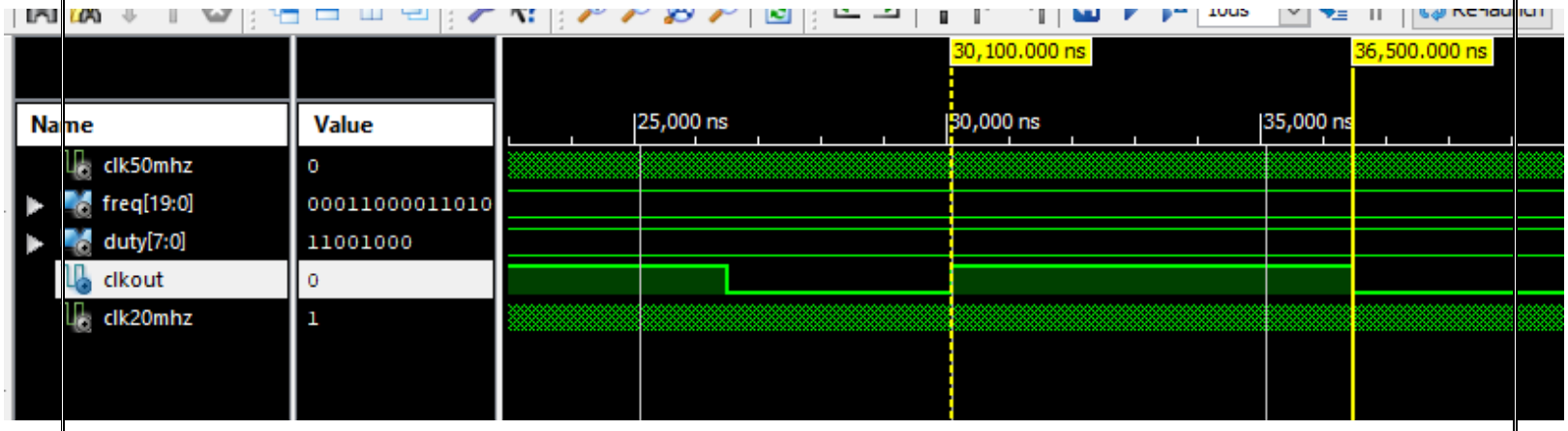
```
-- create duty cycle
if counter * to_integer(unsigned(Freq)) / 256 * to_integer(unsigned(Duty)) >=
10_000_000 then
    clk_state := '0';
end if;
```

همان گونه که از کد واضح است، این کد فقط قابل تست کردن میباشد و قابل سنتز نیست، خروجی ها را به صورت زیر داریم

Test:

1KHz, 200/256 Duty Cycle

```
Freq <= "00011000011010100000";
Duty <= "11001000";
wait for 10 ns;
```



قابل سنتز کردن، Duty Cycle Controller

باید تقسیم را به ضرب تبدیل کنیم، و به این نکته توجه کنیم که اعداد صحیح ما حداکثر 32بیت طول دارند

حال کد زیر را

```
counter * to_integer(unsigned(Freq)) / 256 * to_integer(unsigned(Duty)) >=
10_000_000
```

به

```
counter * to_integer(unsigned(Freq)) * to_integer(unsigned(Duty)) >=
10_000_000 * 256
```

تغییر می دهیم، اما با این مشکل مواجه خواهیم شد که طول اعداد ما از 32 بیت میگذرد.

راه حل؟

برای مثال میتوانیم فرکانس مرجع را تغییر دهیم، تا به این صورت اعداد ما کوچک شوند، البته باید دقت شود که در این صورت دقت Duty Cycle قابل ساخت نیز کم میشود.