Vážení zákazníci,

dovolujeme si Vás upozornit, že na tuto ukázku knihy se vztahují autorská práva, tzv. copyright.

To znamená, že ukázka má sloužit výhradně pro osobní potřebu potenciálního kupujícího (aby čtenář viděl, jakým způsobem je titul zpracován a mohl se také podle tohoto, jako jednoho z parametrů, rozhodnout, zda titul koupí či ne).

Z toho vyplývá, že není dovoleno tuto ukázku jakýmkoliv způsobem dále šířit, veřejně či neveřejně např. umisťováním na datová média, na jiné internetové stránky (ani prostřednictvím odkazů) apod.

redakce nakladatelství BEN – technická literatura
redakce@ben.cz

4. program – běžící světlo

Dalším programem, kde vystačíme s periferií sestávající z řady ledek, je efekt běžícího světla.

Protože tento efekt může mít různou podobu, je třeba upřesnění. Nechť vždy svítí pouze jedna dioda z osmi, přičemž se bude s taktem 0,25 sekundy její poloha přepínat tak, že svítící bod se bude pohybovat zprava doleva. Po dosažení levého okraje se svítící bod skokem přesune na pravou krajní pozici. Pro časování nechť je použit hodinový signál o kmitočtu 50 MHz.

Celý program pak (po vynechání komentářů) vypadá následovně:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity efekt is
    Port (     clk : in STD_LOGIC;
               led : out STD_LOGIC_VECTOR(7 downto 0));
end efekt;

architecture Behavioral of efekt is
signal count  :
      STD_LOGIC_VECTOR(24 downto 0)     := (others => '0');
signal tik    : STD_LOGIC                 := '0';
signal registr   : STD_LOGIC_VECTOR(7 downto 0)
                                          := "00000001";
begin

div:process (clk)
begin
   if clk='1' and clk'event then
      count <= count + 1;
        if count = "10111110101111000011111" then
          count <= (others => '0');
          tik   <= '1';
      else
          tik   <= '0';
      end if;
   end if;
end process;

shift:process (clk)
begin
   if clk='1' and clk'event then
      if tik = '1' then
        registr <= registr(6 downto 0) & registr(7);
      end if;
   end if;
end process;
```

```
led <= registr;

end Behavioral;
```


Entity tohoto programu musí zajistit připojení hodinového signálu (tedy vstup) a výstup na osm led diod. Celý zápis jistě nepotřebuje další komentář.

Architecture tohoto programu je však již oproti předchozím úlohám složitější a tak naopak podrobný rozbor je nutný.

První, co jistě upoutá pozornost, je použití dvou procesů a tří „vnitřních proměnných". Signály count a registr jsou použity samostatně v procesech. Signál tik je použit v obou procesech a slouží k předávání informace z jednoho do druhého. Zde je třeba zdůraznit, že signál může být ovlivňován vždy jen v jednom procesu. Jeho použití, tedy čtení, pak může být v libovolném počtu procesů.

Procesy jsou pro přehlednost pojmenovány. Jméno procesu ukončené dvojtečkou je napsáno před klíčovým slovem process.

První proces, pojmenovaný div, je v principu čítač, tak jak byl vysvětlen v předchozím programu. Uvnitř je však vnořená podmínka, která říká, co se stane, pokud je podmínka splněná a co se stane jinak. Příkaz if byl již v předchozím programu vysvětlen, ale pro jistotu zopakujeme:

```
if podmínka then
    když je podmínka splněná, stane se toto;
else
    jinak se stane něco jiného;
end if;
```

V našem případě pak podmínka říká, co se stane, když hodnota vnitřní proměnné je rovna dané hodnotě, a co se stane, když jí rovna není:

```
if count = "10111110101111000011111" then
     count <= (others => '0');
     tik      <= '1';
else
     tik      <= '0';
end if;
```

V případě splnění podmínky se „vnitřní proměnná" vynuluje a signál tik se nastaví na úroveň log 1. Při nesplnění podmínky se signál tik nastaví na úroveň log 0. Nulováním count vznikl čítač se zkráceným cyklem, tedy čítající jen do dané hodnoty. Signál tik je v hodnotě log 1 právě jen při dosažení nastavené hodnoty a to po dobu jednoho hodinového taktu. Tento signál nám tedy slouží k odměřování určitého časového intervalu. Zbývá vysvětlit, jakým způsobem zjistíme hodnotu signálu count v podmínce.

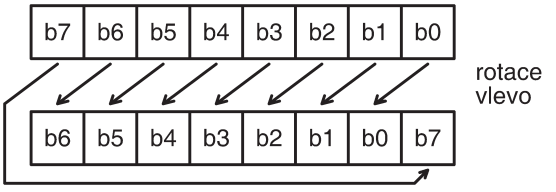
V zadání jsou proto k dispozici dva údaje. Hodinový kmitočet 50 MHz a takt posuvu světla 0,25 s. Tento hodinový signál představuje impulz každých 20 ns. A dále využijeme to, co bylo vysvětleno v předchozí úloze. Abychom získali požadovanou periodu 0,25 s, musíme napočítat 12 500 000 hodinových impulzů. Protože však počítáme od nuly, musíme před-

nastavit hodnotu 12499999. To vyjádřeno v binární soustavě je číslo $101111101011110000011111_b$. S převodem pomůže téměř každá kalkulačka.

Druhý proces, pojmenovaný shift, má ve svém těle podmínku odkazující na stav signál tik. Je-li podmínka splněná, tedy `tik = '1'`, stane se následující:

```
registr <= registr(6 downto 0) & registr(7);
```

Tento zápis říká, že k dolním sedmi bitům proměnné registr se zprava připojí nejvýznamnější bit téže „proměnné“ a uloží se zpět do „proměnné“ registr. Pro názornost je na *obr. 12* grafické vyjádření. Operace se nazývá rotace vlevo.



Obr. 12 Operace rotace vlevo

Pro důkladné vysvětlení tohoto procesu je třeba se vrátit k podmínkám. První je detekce náběžné hrany, tedy s každou náběžnou hranou se kontroluje, jestli je „proměnná“ tik na hodnotě log 1. To se stane každých 0,25 sekundy vždy na jeden hodinový takt. Tak dojde k posuvu právě o jeden bit.

Simulace tohoto programu by jistě byla možná stejným postupem, jaký byl použit v předchozí úloze, ale aby bylo možné ukázat předávání proměnné mezi procesy, musíme si program upravit. „Proměnnou“ tik a stav čítače count vyvedeme na porty. Pro názornost simulace ještě signál count omezíme na čtyři bity a nastavíme, že k nulování dojde při dosažení hodnoty 4. Upravený program pak vypadá takto:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity efekt is
    Port (    clk       : in      STD_LOGIC;
              tik_i     : outSTD_LOGIC;
              count_i   : outSTD_LOGIC_VECTOR(3 downto 0);
              led       : outSTD_LOGIC_VECTOR(7 downto 0));
end efekt;

architecture Behavioral of efekt is
signal count  : STD_LOGIC_VECTOR (3 downto 0):= (others => '0');
signal tik     : STD_LOGIC                    := '0';
signal registr : STD_LOGIC_VECTOR (7 downto 0)    := "00000001";
begin

div:process (clk)
begin
```

```
    if clk='1' and clk'event then
         count <= count + 1;
                   if count = "0100" then
                        count <= (others => '0');
                        tik   <= '1';
                   else
                        tik   <= '0';
                   end if;
    end if;
end process;

shift:process (clk)
begin
    if clk='1' and clk'event then
         if tik = '1' then
           registr <= registr(6 downto 0) & registr(7);
           end if;
    end if;
end process;

led <= registr;
tik_i <= tik;
count_i <= count;

end Behavioral;
```

Simulace tohoto programu znázorněná na *obr. 13* ukazuje, co se děje uvnitř. Čítač div načítá do hodnoty 4. S dalším taktem je tento čítač vynulován a na výstupu tik se objeví log 1 po dobu jednoho hodinového taktu. V souladu s tím, jak byl dříve ukázán postup při výpočtu dělicího poměru čítače, i zde vidíme, že frekvenci signálu tik získáme vydělením frekvence hodinového signálu číslem (n+1), kde n je hodnota přednastavená v podmínce. Z *obr. 13* můžeme pomocí měřicí značky odečíst, že pro periodu hodinového signálu 20 ns a n = 4 je perioda signálu tik 100 ns. To přesně odpovídá předchozímu vysvětlení.

V číslicové technice se takový obvod nazývá kruhový posuvný registr vlevo. Samozřejmě lze realizovat i kruhový posuvný registr vpravo. Zápis této funkce bude:

```
  registr <= registr(0) & registr(7 downto 1);
```

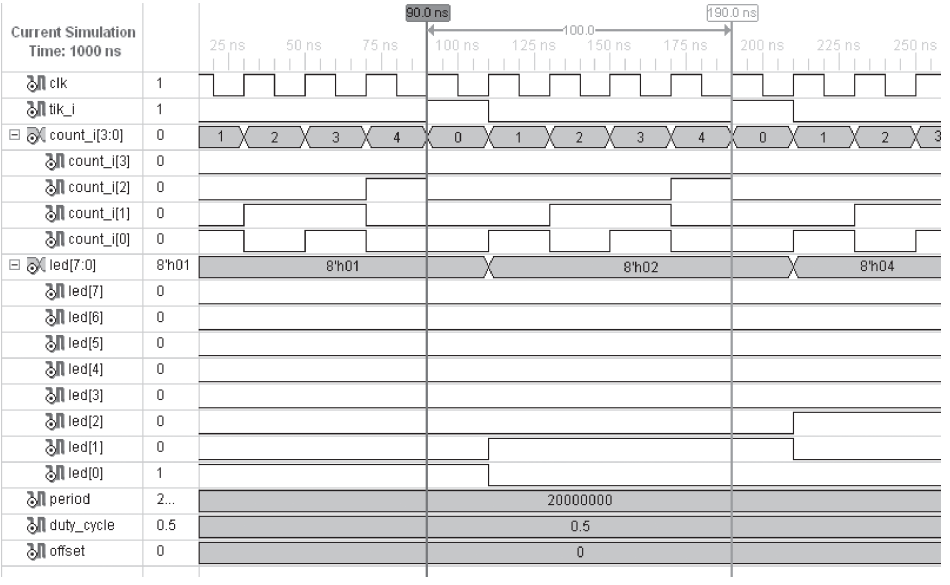
Kromě kruhových posuvných registrů jsou z číslicové techniky známy i posuvné registry vlevo s nasouvající se nulou nebo jedničkou:

```
registr <= registr(6 downto 0) & '0';
registr <= registr(6 downto 0) & '1';
```

Nebo totéž vpravo:

```
registr <= '0' & registr(7 downto 1);
registr <= '1' & registr(7 downto 1);
```

Ve všech těchto zápisech je použit již známý operátor sjednocení – &.



Obr. 13 Simulace kruhového posuvného registru vlevo

I s tímto programem si můžeme pohrát. Například můžeme rozsvítit obě krajní ledky a tyto svítící body necháme s taktem 0,25 s běžet proti sobě.

V architecture se nám změna projeví následujícím. Jednak potřebujeme dva signály, přičemž jedna bude sloužit funkci kruhového registru vlevo a druhá vpravo. A na výstup pak budeme posílat jejich logický součet:

```
signal registr_1 : STD_LOGIC_VECTOR(7 downto 0)  := ”00000001”;
signal registr_2 : STD_LOGIC_VECTOR(7 downto 0)  := ”10000000”;

shift:process (clk)
begin
   if clk=’1’ and clk’event then
      if tik = ’1’ then
         registr_1 <= registr_1(6 downto 0) & registr_1(7);
         registr_2 <= registr_2(0) & registr_2(7 downto 1);
      end if;
   end if;
end process;

led <= registr_1 or registr_2;
```

Abychom mohli spustit program na fyzickém kitu, chybí už jen vytvořit ucf soubor, následně pak konfigurační soubor a ten nahrát.