

Návrh hardware

práce s Xilinx ISE a FITkitem

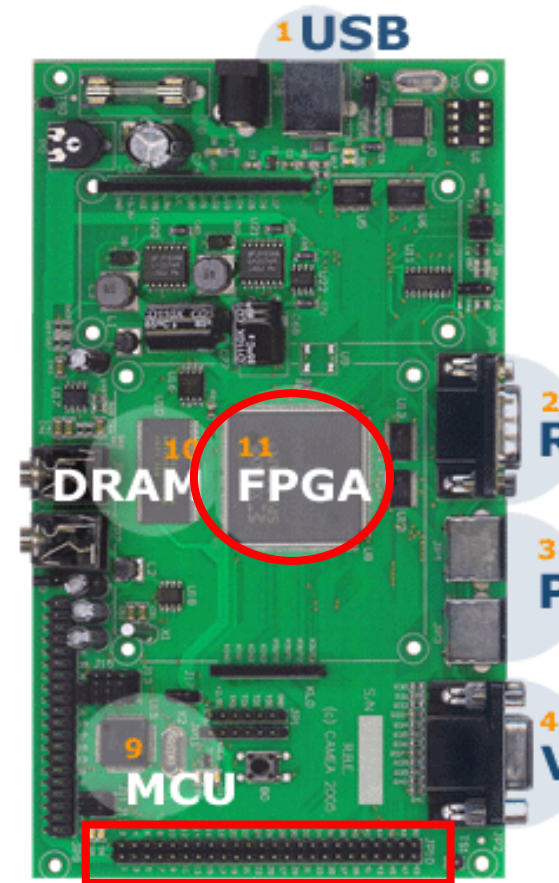
Zdeněk Vašíček, 2016
vasicek@fit.vutbr.cz

Obsah

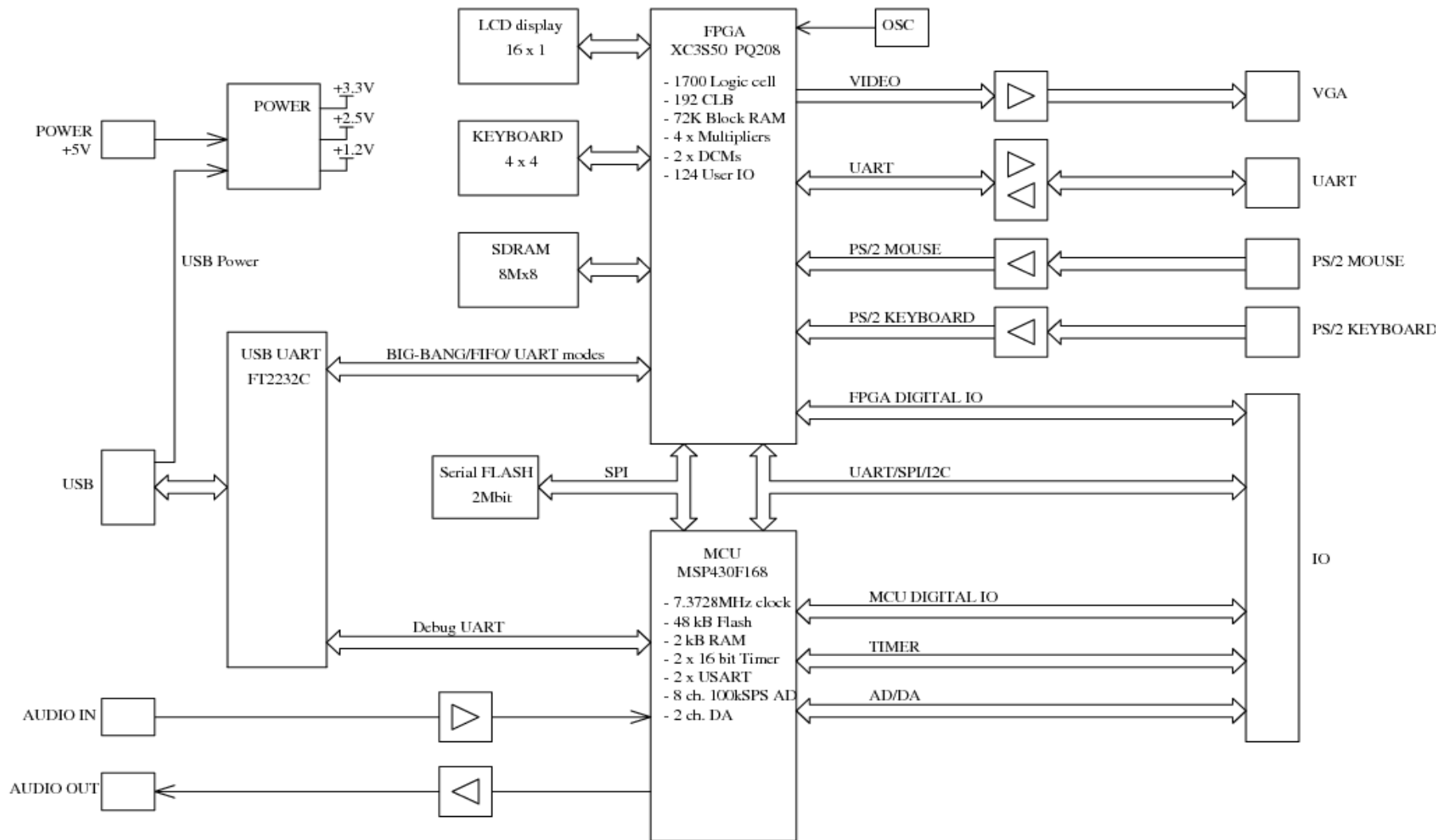
- Xilinx ISE Design Suite
- Využití nástrojů pro FITkit
- Návrh jednoduché aplikace - blikáč

FITkit - FPGA

- FITkit obsahuje dvě programovatelné komponenty
 - mikrokontroler (MCU)
 - FPGA
- **FPGA - čip XC3S50-4PQ208**
 - 1728 logických buněk (LUT4+D), **50k** ekv. log. hradel
 - 2 jednotky DCM pro správu hodin, 4 BRAM paměti (každá 2kB), 4 násobičky 18x18 bitů
 - existuje také verze FITkitu s 8x větším FPGA XC3S400-4PQ208 pro BP/DP
- **Hodinový signál**
 - hardware uvnitř FPGA potřebuje externí hodinový signál – použít signál z MCU označený SMCLK 7.3728MHz
 - uvnitř FPGA lze z SMCLK vyrobit takřka libovolný kmitočet až do 200MHz



FITkit – celkové schema



! Po připojení napájecího napětí je MCU držen v resetu, kit nepracuje

Typický postup při návrhu hardware pomocí Xilinx ISE Design Suite

- Komplexní balík nástrojů pro návrh (ise), syntézu (xst, par, map), simulaci hardware (isim) a práci s constraints (planAhead)
- Typický postup při návrhu hardware pro FPGA v ISE:
 1. vytvořit projekt, zvolit cílovou technologii (v případě FITkitu se jedná o Spartan 3 XC3S50-PQ208)
 2. přidat do projektu zdrojové soubory, v případě nejednoznačnosti zvolit top-level entitu
 3. simulace – vytvořit testbench (volitelně)
 4. vytvořit constraints (mapování pinů, hodinový signál)
 5. syntéza, mapování, generování bitstreamu (pro FITkit bin)
 6. download bitstreamu do FPGA (typicky JTAG)

Čítač připojený na LED

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_unsigned.all;

entity fpga is
    port ( SMCLK : in  STD_LOGIC;
          X : out STD_LOGIC_VECTOR(26 downto 0)
    );
end fpga;

architecture beh of fpga is
    signal cnt: std_logic_vector(23 downto 0) :=
        (others => '0');
begin

    process (SMCLK)
    begin
        if (SMCLK'event and SMCLK = '1') then
            cnt <= cnt + 1;
        end if;
    end process;

    X(26 downto 0) <= (others => 'Z');

    X(8)  <= cnt(23);
    X(10) <= cnt(22);
    X(12) <= cnt(21);

end beh;
```

```
NET "SMCLK"      LOC = "P80" |
PERIOD=8 MHz HIGH 50%;
NET "X<0>"      LOC = "P169" ;
NET "X<1>"      LOC = "P172" ;
NET "X<2>"      LOC = "P176" ;
NET "X<3>"      LOC = "P180" ;
NET "X<4>"      LOC = "P182" ;
NET "X<5>"      LOC = "P184" ;
NET "X<6>"      LOC = "P187" ;
NET "X<7>"      LOC = "P190" ;
NET "X<8>"      LOC = "P194" ;
NET "X<9>"      LOC = "P196" ;
NET "X<10>"     LOC = "P197" ;
NET "X<11>"     LOC = "P198" ;
NET "X<12>"     LOC = "P199" ;
NET "X<13>"     LOC = "P203" ;
NET "X<14>"     LOC = "P204" ;
NET "X<15>"     LOC = "P205" ;
NET "X<16>"     LOC = "P2"   ;
NET "X<17>"     LOC = "P3"   ;
NET "X<18>"     LOC = "P7"   ;
NET "X<19>"     LOC = "P9"   ;
NET "X<20>"     LOC = "P10"  ;
NET "X<21>"     LOC = "P11"  ;
NET "X<22>"     LOC = "P12"  ;
NET "X<23>"     LOC = "P13"  ;
NET "X<24>"     LOC = "P15"  ;
NET "X<25>"     LOC = "P16"  ;
NET "X<26>"     LOC = "P18"  ;
```

Testbench

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

entity tb is
end tb;

architecture beh of tb is

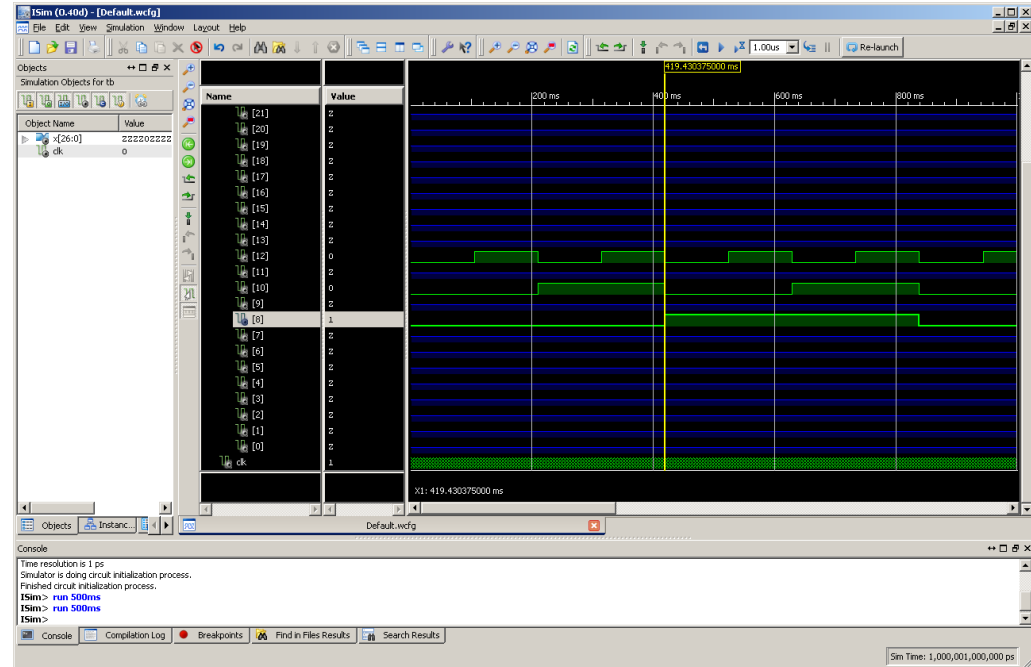
signal x:std_logic_vector(26 downto 0);
signal clk:std_logic := '0';

begin

fpga: entity work.fpga
    port map (SMCLK => clk, x => x);

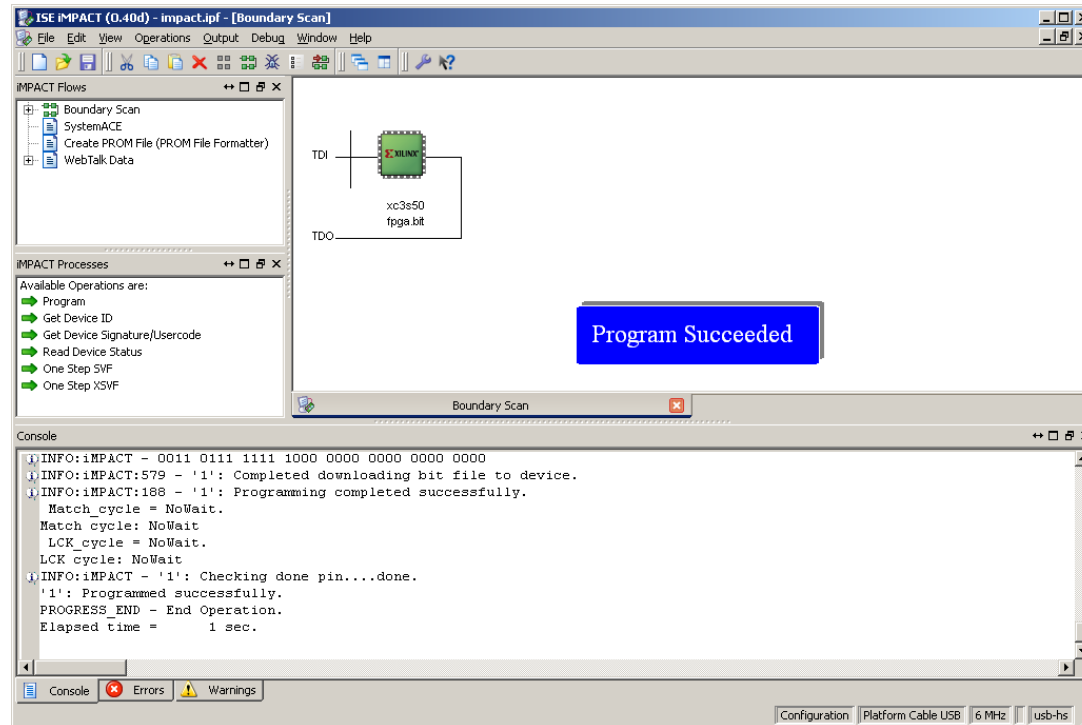
clk <= not clk after 25 ns;

end beh;
```



Programování FPGA na FITkitu

Pomocí JTAG rozhraní (naprogramovat souborem fpga.bit)



Pomocí nástrojů QDevKitu (v příkazové řádce, soubor fpga.bin)

```
fkflash -f testled_f1xx.hex -g testled_f2xx.hex -b fpga.bin
fkterm -d 0
```


Využití nástrojů pro FITkit

- **QDevkit** a další aplikace dovolují automatizovat simulaci, syntézu a programování kitu. Není nutné se přepínat mezi několika nástroji.
- Podpora pro FITkit poskytuje framework pro zjednodušení tvorby aplikací (UCF, top-level, libfitkit).
- Typický postup:
 1. tvorba projektu, definice závislostí, zdrojových souborů a parametrů (soubor project.xml)
 2. tvorba kódu pro MCU a VHDL pro FPGA (např. v ISE)
 3. tvorba simulačního skriptu (ModelSIM nebo ISIM) a testbench souboru pro FPGA (volitelně)
 4. vygenerování Makefile souboru a volání příkazu make (simulace, syntéza, programování)

Textový editor Sublime Text 3

- Archiv dostupný ke stažení na privátních stránkách kurzu obsahuje
 - modifikovaný plugin pro práci s VHDL soubory,
 - překladový skript pro FITkit projekt,
 - plugin pro tvorbu nového projektu pro FITkit

Práce s projekty:

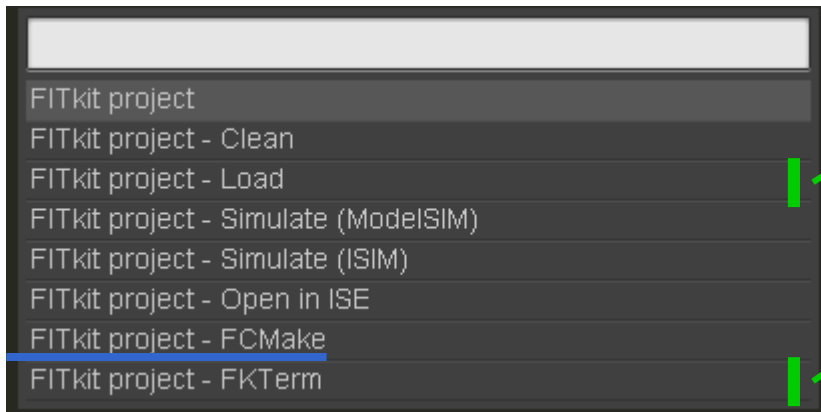
- **Otevření existující složky:**
Project > Add Folder to Project > zvolit např. FITKITSVN/apps/demo/led
- **Vytvoření nové složky s projektem:**
Project > Create Project > FITkit project

Pozor, nová složka musí být umístěna v podadresáři FITKITSVN, např. apps/ivh/test

Textový editor Sublime Text 3

Překlad projektu

- Nastavit výchozí volbu pomocí Tools > Build system > FITkit
- Následně lze používat Tools > Build with ... **Ctrl+Shift+B**



Přeloží (došlo-li ke změně) a nahraje projekt do FITkitu

Spustí terminál (aplikaci na kitu)

- Aby bylo možné používat překladový systém, musí se před prvním použitím vygenerovat Makefile pomocí **FCMake !!**

Textový editor Sublime Text 3

Užitečné tipy pro tvorbu VHDL kódu

- `spro` + `<TAB>` + `name` + `<TAB>` + `<TAB>` – synchronní proces `name`
- `apro` + `<TAB>` – synchronní proces s asynchronním nulováním
- `sigl` + `<TAB>` – deklarace signálu typu `std_logic`
- `sigsl` + `<TAB>` – deklarace signálu typu `std_logic_vektor`
- `vhdl` + `<TAB>` – entita a architektura včetně hlaviček
- `entarch` + `<TAB>` – entita a architektura
- `typefsm` + `<TAB>` – deklarace datového typu a signálů pro FSM
- `fsm` + `<TAB>` – proces pro aktuální a následující stav FSM
- `forg` + `<TAB>` – konstrukce for-generate

- Další rychlé zkratky lze definovat vytvořením šablony v adresáři `Data\Packages\User\VHDL\Snippets\`

Projekt pro FITkit

project.xml

```
<?xml version="1.0" encoding="utf-8"?>
<project outputprefix="testled">

  <name>LED 1</name>

  <!-- MCU part -->
  <mcu>
    <file>main.c</file>
  </mcu>

  <!-- FPGA part -->
  <fpga architecture="gp"
dcmfrequency="20MHz">
    <file>top.vhd</file>
  </fpga>

</project>
```

fpga/sim/isim.tcl

```
#Project setup
#=====
#set TESTBENCH_ENTITY "testbench"
#set ISIM_PRECISION "1 ps"

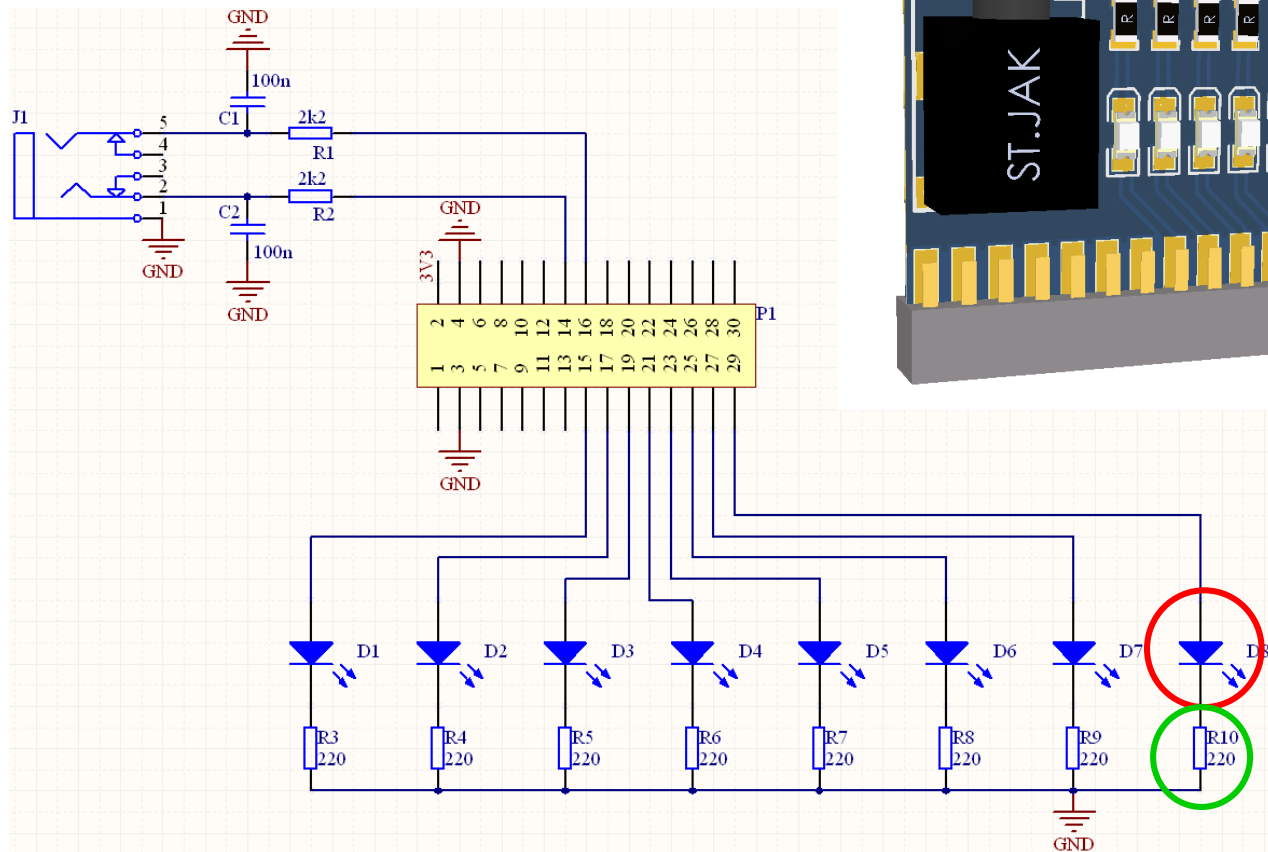
#Run simulation
#=====
proc isim_script {} {

    add_divider "Top level FPGA"
    add_wave_label "-color #FFFF00"
                        "reset" /testbench/uut/rst

    run 200 ms
}
```

PWM modul

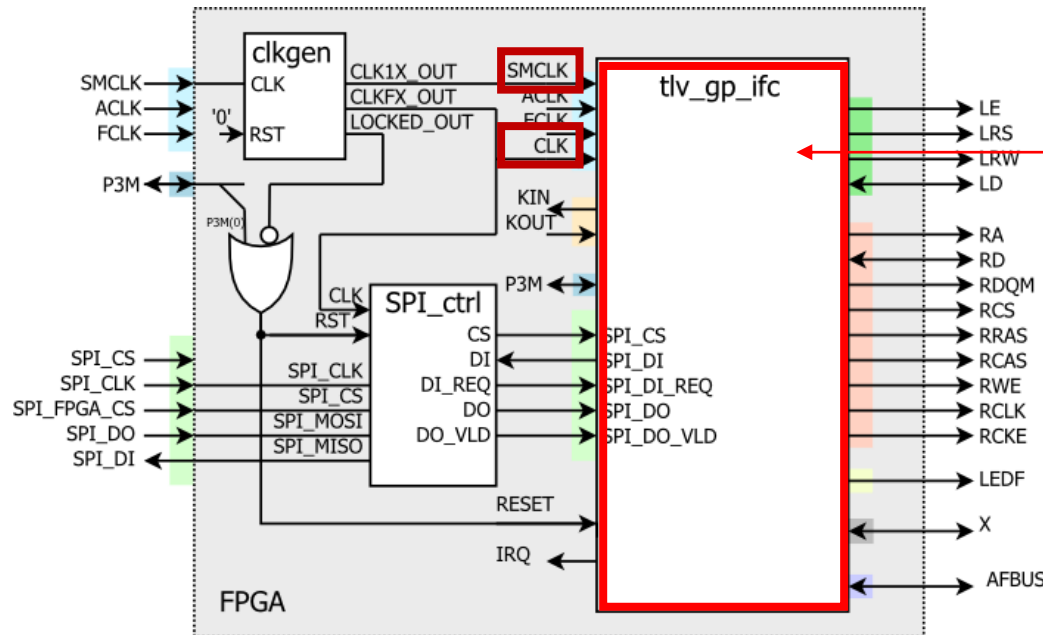
- LED diody se aktivují hodnotou log. 1



Nastavení projektu

vzhledem ke způsobu připojení PWM modulu k FITkitu

- Použijeme předdefinovanou entitu `tlv_gp_ifc`, která obsahuje port pojmenovaný `X`, na který máme připojeny LED diody (spodní konektor)



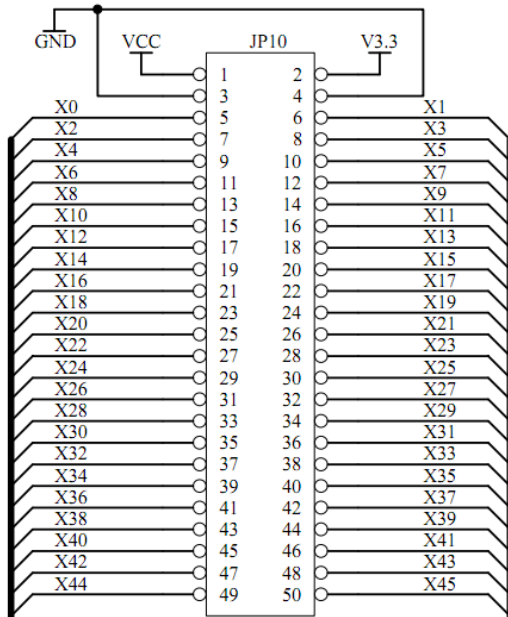
Ve VHDL popisujeme v rámci projektu pouze architekturu, entita je předdefinována v frameworku FITkitu.

- V souboru `project.xml` je nutné v sekci `FPGA` zvolit `architecture="gp"`
- Spodní konektor JP10 je namapován na port `X` entity `tlv_gp_ifc`.
- Synchronní chování lze odvodit od hodinového signálu `SMCLK` (pevný kmitočet) nebo `CLK` (kmitočet lze měnit v `project.xml`)

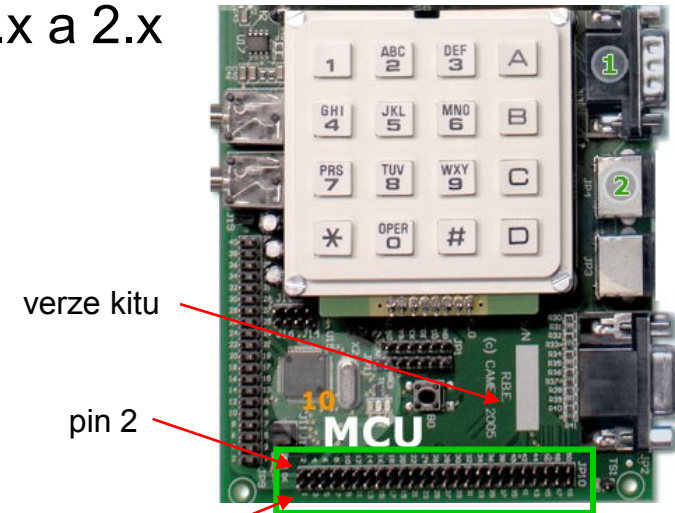
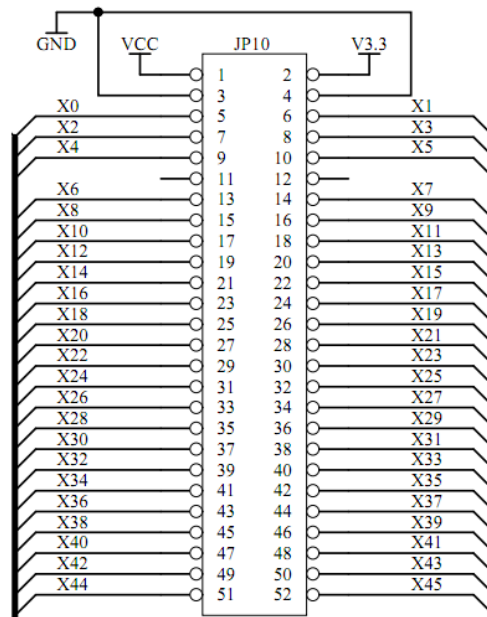
Konektor JP10 a jeho mapování na signál X dostupný v FPGA

- Zapojení konektoru se liší pro FITkit verze 1.x a 2.x

FITkit verze 1.x

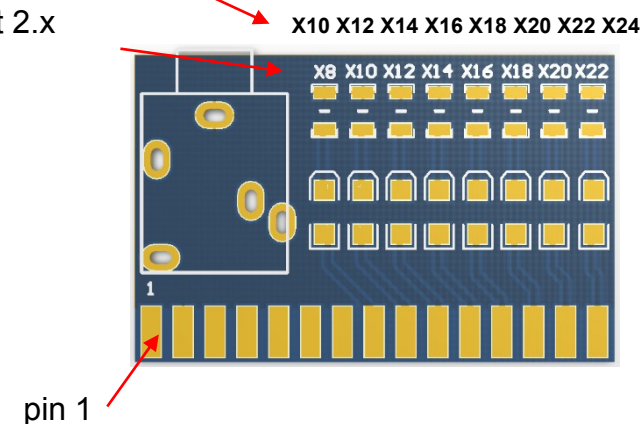


FITkit verze 2.x





FITkit 1.x (+2)

FITkit 2.x



Rozsvícení diod

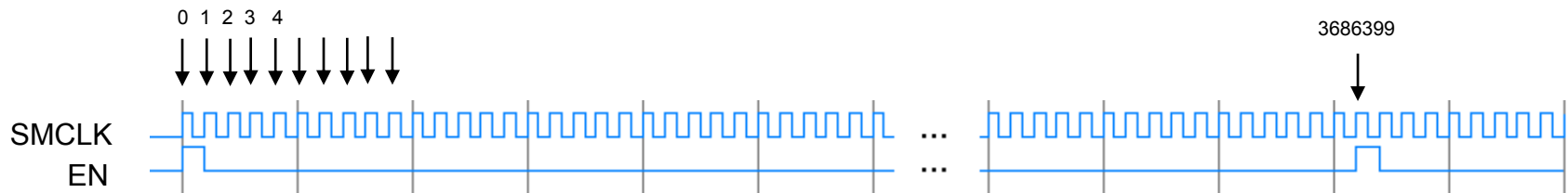
```
architecture beh of tlv_gp_ifc is
    signal led : std_logic_vector(7 downto 0) := "10000001"; -- vychozi stav
begin
    
    x(8) <= led(0); x(10) <= led(1); x(12) <= led(2); ...

    
    x(10) <= led(0); x(12) <= led(1); x(14) <= led(2); ...
end architecture;
```

```
led_con: for i in 0 to 7 generate
    x(8 + 2*i) <= leds(i); -- FITkit 2.x
    -- x(10 + 2*i) <= leds(i); -- FITkit 1.x
end generate;
```

Tvorba povolovacího signálu o frekvenci 2Hz

- Vstup: hodinový signál s kmitočtem 7.3728 MHz (SMCLK)
- Výstup: signál aktivní jeden hodinový takt dvakrát za sekundu



- Povolovací signál lze získat tak, že budeme počítat vzestupné hrany a jakmile napočítáme do 3686399 ($7372800 / 2 - 1$), nastavíme hodnotu EN na 1 a vynulujeme čítač.
- Šířka čítače:
 - $2^W \geq 7372800/2 \rightarrow W = \lceil \log_2 3686400 \rceil = 22 \text{ bitů}$

Tvorba povolovacího signálu o frekvenci 2Hz

```
library IEEE;
use IEEE.std_logic_1164.ALL; use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

architecture beh of tlv_gp_ifc is
    subtype cntt is integer range 0 to 7372800/2-1;
    signal cnt : cntt := 0;
    signal en : std_logic := 0;
begin

    process (SMCLK)
    begin
        if (SMCLK'event and SMCLK = '1') then
            en <= '0';
            if (cnt = cntt'HIGH) then
                cnt <= 0;
                en <= '1';
            else
                cnt <= cnt + 1;
            end if;
        end if;
    end process;

    ...
end architecture;
```

Tvorba povolovacího signálu o frekvenci 2Hz

```
library IEEE;
use IEEE.std_logic_1164.ALL; use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

architecture beh of tlv_gp_ifc is
    signal cnt: std_logic_vector(21 downto 0) := (others => '0');
    signal cnt_lst: std_logic;
    signal en : std_logic;
begin

    process (SMCLK)
    begin
        if (SMCLK'event and SMCLK = '1') then
            if (cnt_lst = '1') then
                cnt <= (others => '0');
            else
                cnt <= cnt + 1;
            end if;
        end if;
    end process;

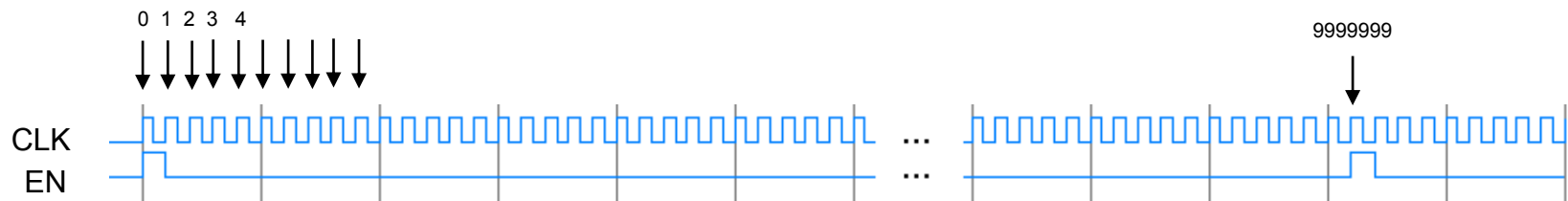
    cnt_lst <= '1' when cnt = conv_std_logic_vector(7372800/2 - 1, 22) else '0';

    en <= cnt_lst;

end beh;
```

Tvorba povolovacího signálu o frekvenci 2Hz

- Vstup: hodinový signál s kmitočtem 20 MHz (CLK)
- Výstup: signál aktivní jeden hodinový takt dvakrát za sekundu



- Povolovací signál lze získat tak, že budeme počítat vzestupné hrany a jakmile napočítáme do 9999999, nastavíme hodnotu EN na 1 a vynulujeme čítač.
- 20MHz signál lze získat tak, že v project.xml přidáme do sekce <fpga> atribut dcmfrequency s hodnotou "20MHz"


```
<!-- FPGA part -->  
<fpga architecture="gp" dcmfrequency="20MHz">  
  <file>top.vhd</file>  
</fpga>
```

Blikač s periodou jedna sekunda


- Pro blikání s periodou jedna sekunda musíme 2x za sekundu invertovat stav registru, který je připojen na diodu.
- Můžeme využít toho, že umíme generovat povolovací signál s frekvencí 2 Hz a ten připojíme na EN vstup registru.

```
architecture beh of tlv_gp_ifc is
    ...
    signal led : std_logic := '0'; -- vychozi stav
begin

    process (SMCLK)
    begin
        if (SMCLK'event and SMCLK = '1') then
            if (en = '1') then
                led <= not led;
            end if;
        end if;
    end process;

    
    FITkit 2.x
    x(8) <= led; x(22) <= not led;

    nebo

    
    FITkit 1.x
    x(10) <= led; x(24) <= not led;

    ...
end;
```

Testbench

- Všechny VHDL soubory v adresáři [fpga/sim](#) jsou považovány za zdrojové soubory pro simulaci.
- Stačí vytvořit např. `tb.vhd`, uvnitř souboru instancovat entitu, kterou chceme otestovat
 - `work.fpga`,
 - `work.tlv_gp_ifc`
 - nebo jinou
- Testbench souborů může být několik, výběr konkrétní testbench entity se provádí v simulačním skriptu [fpga/sim/isim.tcl](#) pomocí proměnné `TESTBENCH_ENTITY`

```
#Project setup
#=====
#set ISIM_PRECISION "1 ps"

if (0) {
    set TESTBENCH_ENTITY "testbench_fpga"
} else {
    set TESTBENCH_ENTITY "testbench_gp"
}

#Run simulation
#=====
proc isim_script {} {

    add_divider "TB"
    wave add /

    add_divider "LED"
    add_wave_label "" "X" /xbus

    run 100 ms
}
```

Světelný had

posuvný registr (rotace)

```
architecture beh of tlv_gp_ifc is
    ...
    signal leds: std_logic_vector(7 downto 0) :=
        (0 => '1', others => '0');
begin

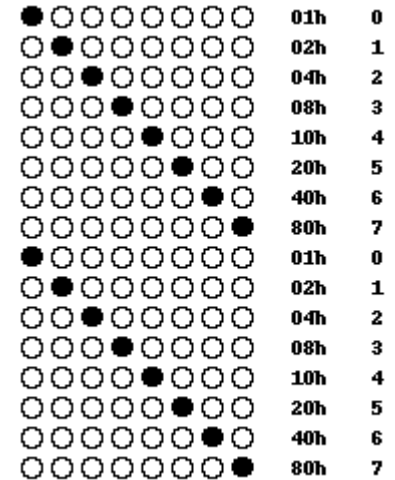
    process (SMCLK)
    begin
        if (SMCLK'event and SMCLK = '1') then
            if (en = '1') then
                leds <= leds(6 downto 0) & leds(7);
            end if;
        end if;
    end process;

    led_con: for i in 0 to 7 generate
        X(8 + 2*i) <= leds(i); -- FITkit 2.x
        -- X(10 + 2*i) <= leds(i); -- FITkit 1.x

    end generate;

    ...

end beh;
```

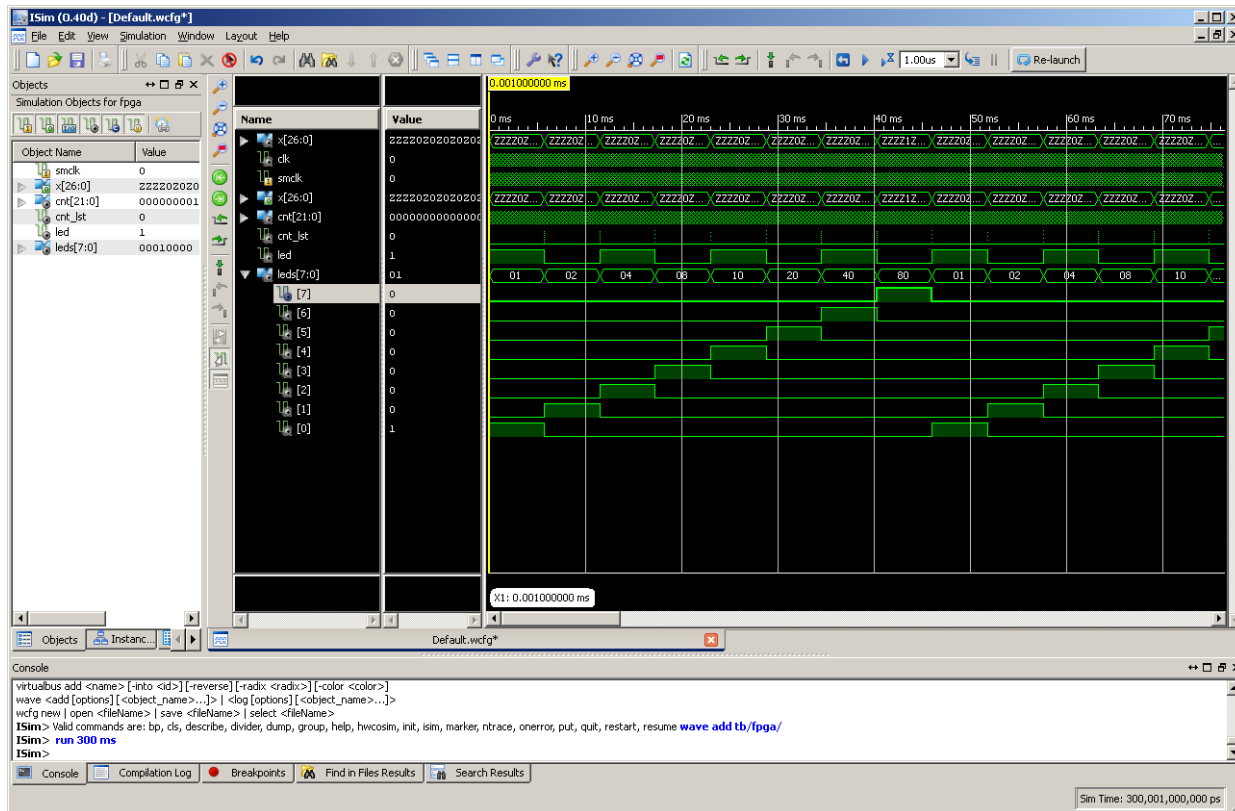


Světelný had

Ukázka simulace pro

```
cnt_1st <= '1' when cnt = conv_std_logic_vector(7372800/64, 22) else '0';
```

```
wave add /tb/fpga/
```



Světelný had – 2

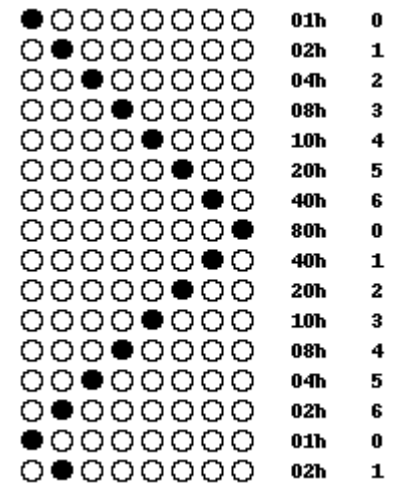
posuvný registr (doleva / doprava), registr pro směr,
čítač 3 bitový modulo 7, komparátor

```
signal led_cnt : std_logic_vector(2 downto 0) := "000";
signal led_dir : std_logic := '0';

signal led_cnt_rst : std_logic := '0';
signal led_cnt_last : std_logic;

process (SMCLK)
begin
    if (SMCLK'event and SMCLK='1') then
        if (led_cnt_rst = '1') then
            led_cnt <= (others => '0');
        elsif (en = '1') then
            led_cnt <= led_cnt + 1;
        end if;
    end if;
end process;

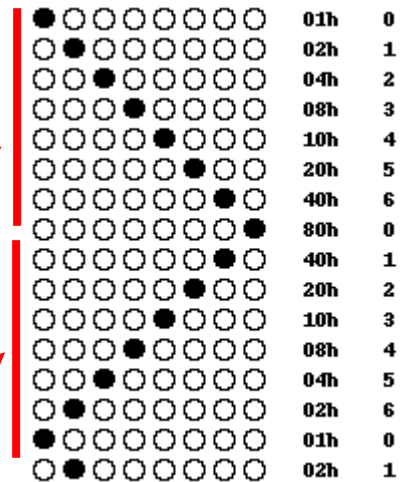
led_cnt_last <= '1' when led_cnt = "110" else '0';
```



Světelný had – 2

posuvný registr (doleva / doprava), registr pro směr,
čítač 3bitový modulo 7, komparátor

```
process (SMCLK)
begin
  if (SMCLK'event and SMCLK='1') then
    led_cnt_rst <= '0';
    if (en = '1') then
      if (led_dir = '0') then --doleva
        leds <= leds(6 downto 0) & '0';
        if (led_cnt_last = '1') then
          led_dir <= '1';
          led_cnt_rst <= '1';
        end if;
      else --doprava
        leds <= '0' & leds(7 downto 1);
        if (led_cnt_last = '1') then
          led_dir <= '0';
          led_cnt_rst <= '1';
        end if;
      end if;
    end if;
  end if;
end if;
end process;
```



Světelný had – 3

● ○ ○ ○ ○ ○ ○ ○ ○ ○	01h	0
○ ● ○ ○ ○ ○ ○ ○ ○ ○	02h	1
○ ○ ● ○ ○ ○ ○ ○ ○ ○	04h	2
○ ○ ○ ● ○ ○ ○ ○ ○ ○	08h	3
○ ○ ○ ● ○ ○ ○ ○ ○ ○	04h	0
○ ● ○ ○ ○ ○ ○ ○ ○ ○	02h	1
● ○ ○ ○ ○ ○ ○ ○ ○ ○	01h	2
○ ● ○ ○ ○ ○ ○ ○ ○ ○	02h	0
○ ○ ● ○ ○ ○ ○ ○ ○ ○	04h	1
○ ○ ○ ● ○ ○ ○ ○ ○ ○	08h	2
○ ○ ○ ○ ● ○ ○ ○ ○ ○	10h	3
○ ○ ○ ○ ○ ● ○ ○ ○ ○	20h	4
○ ○ ○ ○ ○ ○ ● ○ ○ ○	40h	5
○ ○ ○ ○ ○ ○ ○ ● ○ ○	80h	6
○ ○ ○ ○ ○ ○ ○ ● ○ ○	40h	0
○ ○ ○ ○ ○ ○ ○ ● ○ ○	20h	1
○ ○ ○ ○ ○ ● ○ ○ ○ ○	10h	2
○ ○ ○ ○ ● ○ ○ ○ ○ ○	08h	3
○ ○ ○ ● ○ ○ ○ ○ ○ ○	04h	4
○ ○ ● ○ ○ ○ ○ ○ ○ ○	02h	5
● ○ ○ ○ ○ ○ ○ ○ ○ ○	01h	0

Světelný had – 3

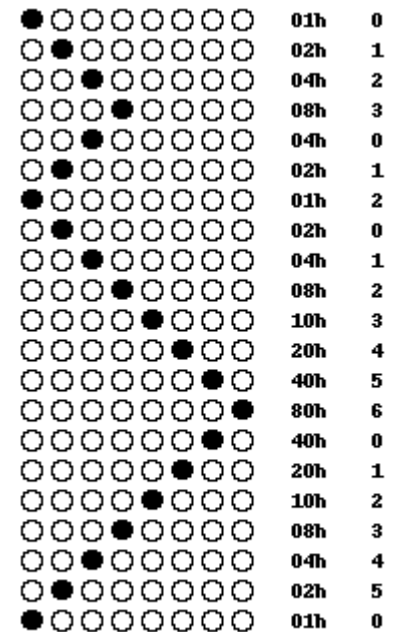
posuvný registr (doleva / doprava), čítač 3bitový,
čtyři komparátory, automat - 4 stavy

●○○○○○○○○	01h	0
○●○○○○○○○	02h	1
○○●○○○○○○	04h	2
○○○●○○○○○	08h	3
○○○○●○○○○	04h	0
○○○○○●○○○	02h	1
○○○○○○●○○	01h	2
○○○○○○○●○	02h	0
○○○○○○○○●	04h	1
○○○○○○○○○	08h	2
○○○○○○○○○	10h	3
○○○○○○○○○	20h	4
○○○○○○○○○	40h	5
○○○○○○○○○	80h	6
○○○○○○○○○	40h	0
○○○○○○○○○	20h	1
○○○○○○○○○	10h	2
○○○○○○○○○	08h	3
○○○○○○○○○	04h	4
○○○○○○○○○	02h	5
○○○○○○○○○	01h	0

Světelný had – 3

posuvný registr (doleva / doprava), čítač 3bitový,
čtyři komparátory, automat - 4 stavy

```
signal leds : std_logic_vector(7 downto 0) :=  
                                (0 => '1',others => '0');  
signal led_updir : std_logic := '0';  
  
process (SMCLK)  
begin  
    if (SMCLK'event and SMCLK='1') then  
        if (en = '1') then  
            if (led_updir = '1') then  
                leds <= leds(6 downto 0) & '0';  
            else  
                leds <= '0' & leds(7 downto 1);  
            end if;  
        end if;  
    end if;  
end process;
```



Světelný had – 3

posuvný registr (doleva / doprava), čítač 3bitový,
čtyři komparátory, automat - 4 stavy

```
signal led_cnt : std_logic_vector(2 downto 0) := "000";
signal led_cnt_rst : std_logic := '0';
signal led_cnt_2, led_cnt_3, led_cnt_5, led_cnt_6 : std_logic;

process (SMCLK)
begin
    if (SMCLK'event and SMCLK='1') then
        if (en = '1') then
            if (led_cnt_rst = '1') then
                led_cnt <= (others => '0');
            else
                led_cnt <= led_cnt + 1;
            end if;
        end if;
    end if;
end process;

led_cnt_2 <= '1' when led_cnt = "010" else '0';
led_cnt_3 <= '1' when led_cnt = "011" else '0';
led_cnt_5 <= '1' when led_cnt = "101" else '0';
led_cnt_6 <= '1' when led_cnt = "110" else '0';
```

Světelný had – 3

posuvný registr (doleva / doprava), čítač 3bitový,
čtyři komparátory, **automat - 4 stavy**

```
type fsmstate is (s0,s1,s2,s3);
signal pstate: fsmstate := s0;
signal nstate: fsmstate;

process (SMCLK)
begin
    if (SMCLK'event and SMCLK='1') then
        if (en = '1') then pstate <= nstate; end if;
    end if;
end process;

process (pstate, led_cnt_2, led_cnt_3, led_cnt_5, led_cnt_6)
begin
    nstate <= s0;
    led_cnt_rst <= '0';
    led_updir <= '0';
    case pstate is
        when s0 =>
            led_updir <= '1';
            if (led_cnt_3 = '1') then
                nstate <= s1;
                led_cnt_rst <= '1';
            end if;
    end case;
end process;
```


Světelný had – 3

posuvný registr (doleva / doprava), čítač 3bitový,
čtyři komparátory, **automat - 4 stavy**

```
when s1 =>
    nstate <= s1;
    if (led_cnt_2 = '1') then
        nstate <= s2;
        led_updir <= '1';
        led_cnt_rst <= '1';
    end if;
when s2 =>
    nstate <= s2;
    led_updir <= '1';
    if (led_cnt_6 = '1') then
        nstate <= s3;
        led_cnt_rst <= '1';
    end if;
when s3 =>
    nstate <= s3;
    if (led_cnt_5 = '1') then
        nstate <= s0;
        led_cnt_rst <= '1';
        led_updir <= '1';
    end if;
end case;
end process;
```

Zobrazovač znaků

(nepravidelný vzor)

čítač, paměť ROM (dekodér)

● ● ● ● ● ● ○ ○	3Fh	0
○ ○ ● ○ ○ ○ ● ○	44h	1
○ ○ ● ○ ○ ○ ○ ●	84h	2
○ ○ ● ○ ○ ○ ○ ●	84h	3
○ ○ ● ○ ○ ○ ● ○	44h	4
● ● ● ● ● ● ○ ○	3Fh	5
○ ○ ○ ○ ○ ○ ○ ○	00h	6
○ ○ ○ ○ ○ ○ ○ ○	00h	7

```
signal line_cnt : std_logic_vector(2 downto 0) := "000";
type trom is array (0 to 7) of std_logic_vector(7 downto 0);
signal rom : trom :=
    (X"3F", X"44", X"84", X"84", X"44", X"3F", X"00", X"00");

process (SMCLK)
begin
    if (SMCLK'event and SMCLK='1') then
        if (en = '1') then
            line_cnt <= line_cnt + 1;
        end if;
    end if;
end process;

leds <= rom(conv_integer(line_cnt));
```

Chyby

```
process (SMCLK)
begin
  if (SMCLK'event and SMCLK='1') then
    led_cnt <= (others => '0');
  elsif (en = '1') then
    led_cnt <= led_cnt + 1;
  end if;
end process;
```

```
process (SMCLK)
begin
  if (SMCLK'event and SMCLK='1') then
    if (en'event and en = '1') then
      led_cnt <= led_cnt + 1;
    end if;
  end if;
end process;
```

```
process (SMCLK)
begin
  ...
  led <= '1';
  ...
end process;

process (SMCLK)
begin
  ...
  led <= '0';
  ...
end process;
```