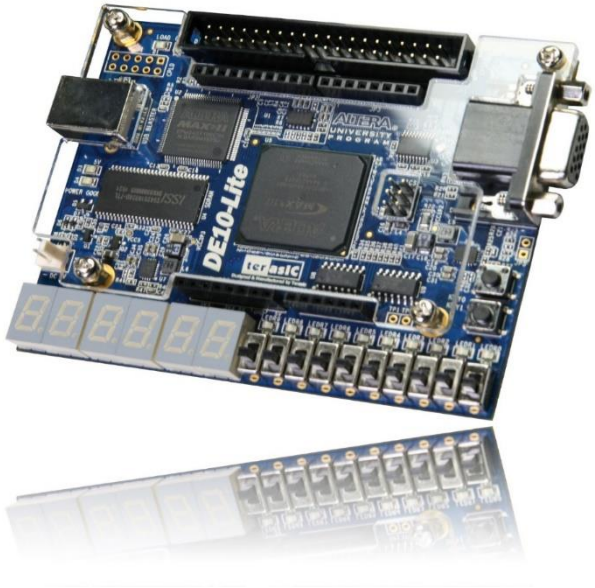
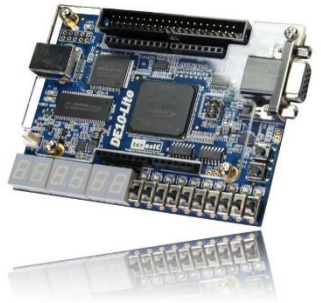


GRAFCET電路設計-呼吸燈

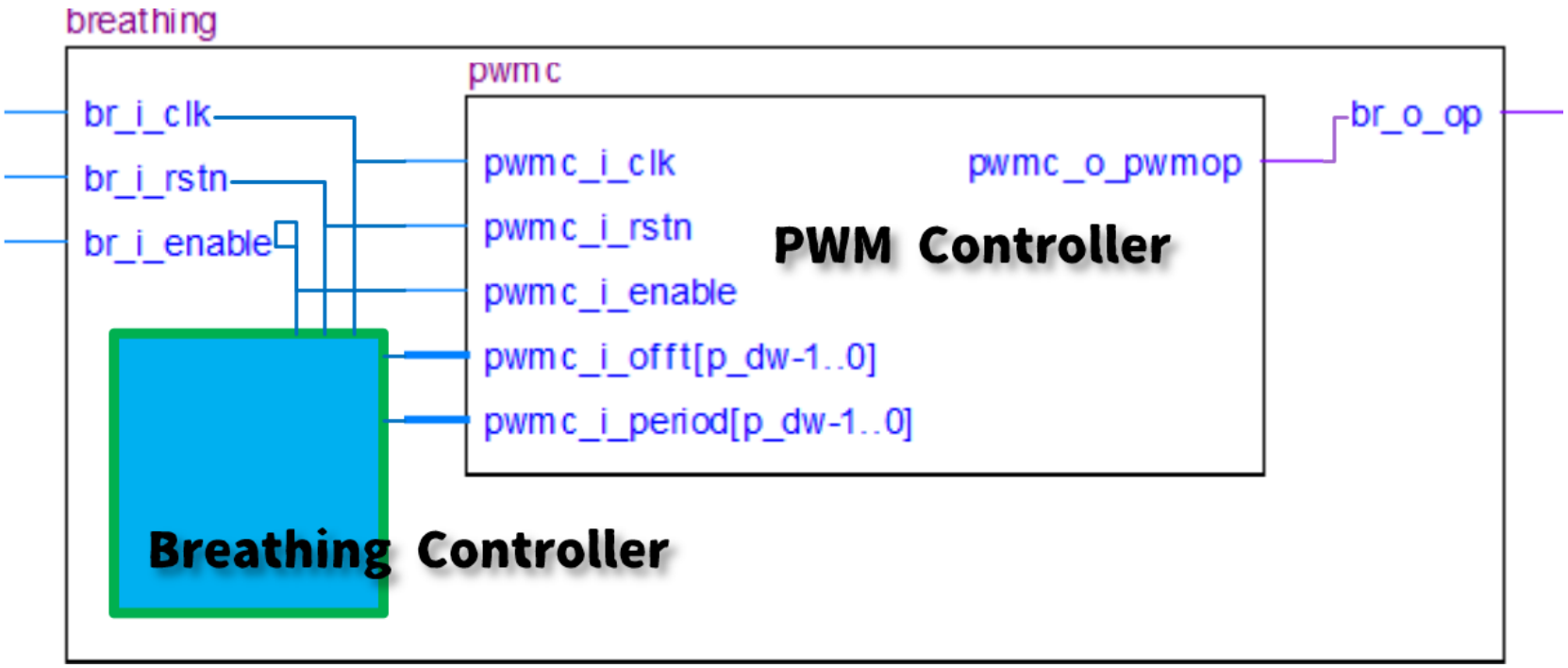
CSIE MIAT Lab.

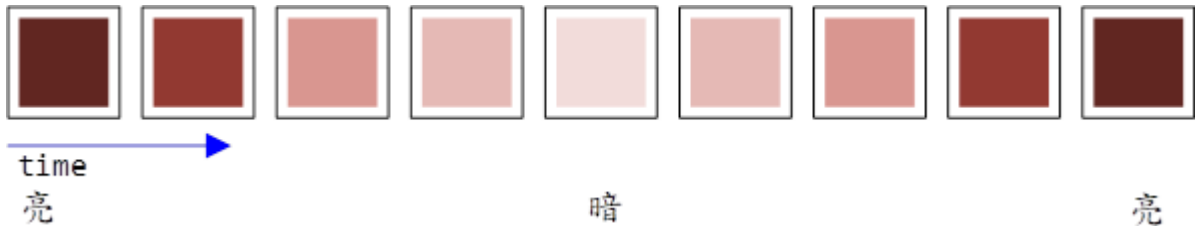
Outline

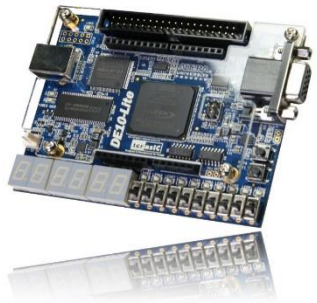
- 呼吸燈
- PWM控制器
- 呼吸燈控制器
- 隨堂練習



呼吸燈


breathing
pwmc
br_i_clk
br_i_rstn
br_i_enable
pwmc_i_clk
pwmc_i_rstn
pwmc_i_enable
pwmc_i_offt[p_dw-1..0]
pwmc_i_period[p_dw-1..0]
pwmc_o_pwmop
br_o_op
PWM Controller
Breathing Controller

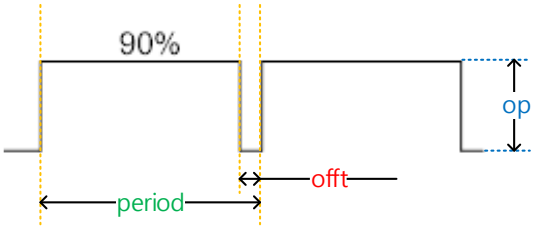

time
亮
暗
亮



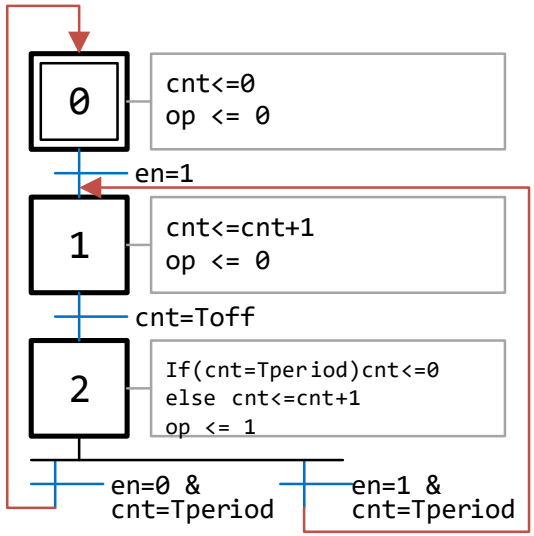
PWM控制器

- Interface

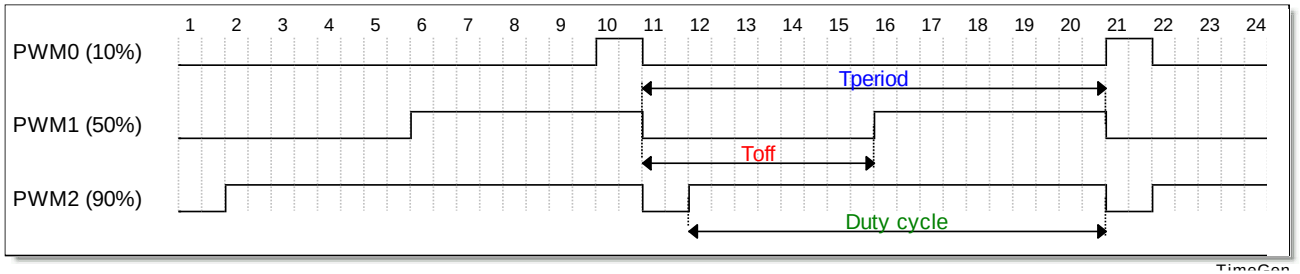
Sig.	Dir.	Bit	Desc.
clk	in	1	時脈輸入
rstn	in	1	重置
en	in	1	致能
offt	in	8 (Default)	如圖
period	in	8 (Default)	如圖
op	out	1	如圖

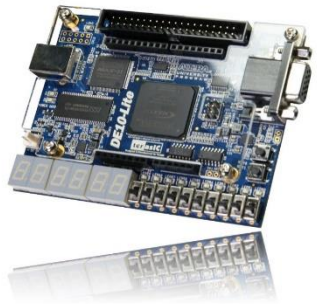


- Grafect離散事件



- Waveform





PWM控制器

- VHDL Source Code
  - pwmc.vhdl

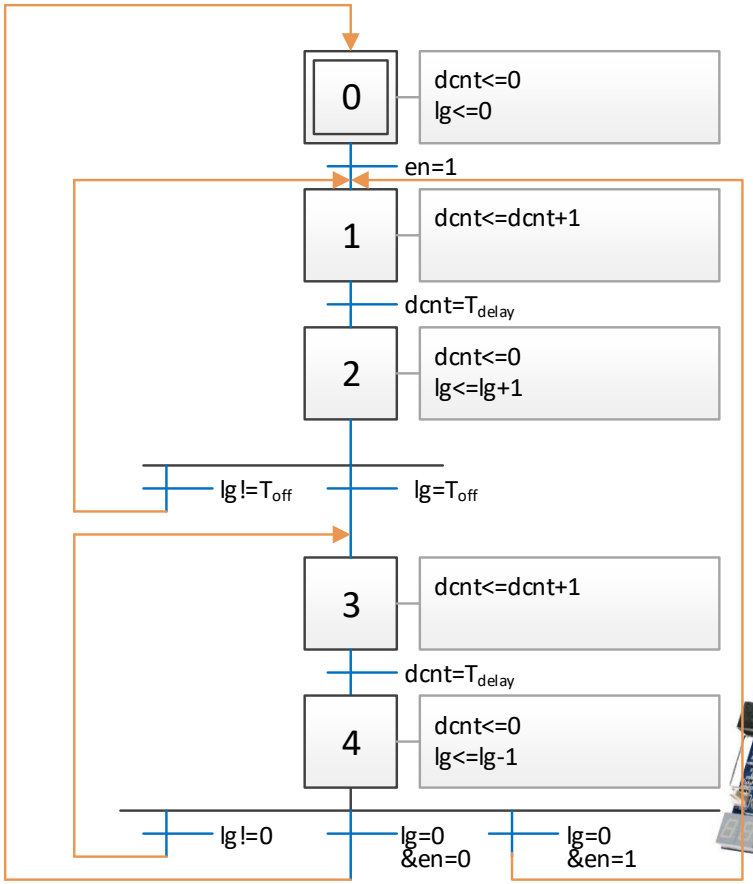
```vhdl
library ieee;
    use ieee.std_logic_1164.all;
    use ieee.std_logic_unsigned.all;

entity pwmc is
    generic(
        p_dw : integer := 8
    );
    port(
        pwmc_i_clk    : in  std_logic;
        pwmc_i_rstn   : in  std_logic;
        pwmc_i_enable : in  std_logic;
        pwmc_i_offt   : in  std_logic_vector((p_dw-1) downto 0);
        pwmc_i_period : in  std_logic_vector((p_dw-1) downto 0);
        pwmc_o_pwmop  : out std_logic
    );
end entity pwmc;

architecture rtl of pwmc is

    type grafect is (X0, X1, X2);
    signal w_next_state : grafect;
    signal r_curr_state : grafect;

    signal w_cntr : std_logic_vector((p_dw-1) downto 0);
    signal r_cntr : std_logic_vector((p_dw-1) downto 0);

    signal w_pwmbuf : std_logic;
    signal r_pwmbuf : std_logic;

begin

    gstate: process(pwmc_i_clk, pwmc_i_rstn)
    begin
        if pwmc_i_rstn = '0' then
            r_curr_state <= X0;
        elsif rising_edge(pwmc_i_clk) then
            r_curr_state <= w_next_state;
        end if;
    end process;

    control_path: process(r_curr_state, pwmc_i_enable, r_cntr, pwmc_i_offt, pwmc_i_period)
    begin
        case r_curr_state is
            when X0 =>
                if pwmc_i_enable = '1' then
                    w_next_state <= X1;
                else
                    w_next_state <= X0;
                end if;
            when X1 =>
                if r_cntr = pwmc_i_offt then
                    w_next_state <= X2;
                else
                    w_next_state <= X1;
                end if;
            when X2 =>
                if r_cntr = pwmc_i_period then
                    if pwmc_i_enable = '1' then
                        w_next_state <= X1;
                    else
                        w_next_state <= X0;
                    end if;
                else
                    w_next_state <= X2;
                end if;
        end case;
    end process;

    data_path: process(r_curr_state, r_cntr, pwmc_i_offt, pwmc_i_period)
    begin
        case r_curr_state is
            when X0 =>
                w_cntr <= (others => '0');
                w_pwmbuf <= '0';
            when X1 =>
                w_cntr <= r_cntr + '1';
                w_pwmbuf <= '0';
            when X2 =>
                if r_cntr = pwmc_i_period then
                    w_cntr <= (others => '0');
                else
                    w_cntr <= r_cntr + '1';
                end if;
                w_pwmbuf <= '1';
        end case;
    end process;


    outbuf_path: process(pwmc_i_clk, pwmc_i_rstn)
    begin
        if pwmc_i_rstn = '0' then
            r_cntr <= (others => '0');
            r_pwmbuf <= '0';
        elsif rising_edge(pwmc_i_clk) then
            r_cntr <= w_cntr;
            r_pwmbuf <= w_pwmbuf;
        end if;
    end process;

    pwmc_o_pwmop <= r_pwmbuf;

end rtl;
```

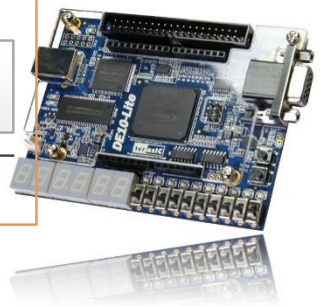
呼吸燈控制器

- Interface

Sig.	Dir.	Bit	Desc.
clk	in	1	時脈輸入
rstn	in	1	重置
en	in	1	致能
op	out	1	輸出

- Grafect離散事件





呼吸燈控制器

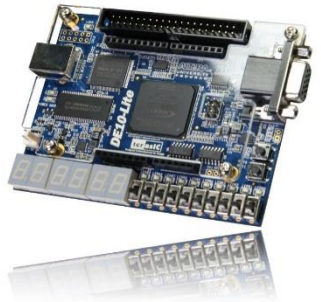
- VHDL Source Code
  - breathing.vhdl

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

entity breathing is
    generic(
        lg_dw : integer := 8;
        dcnt_dw : integer := 16
    );
    port(
        br_i_clk    : in  std_logic;
        br_i_rstn   : in  std_logic;
        br_i_enable : in  std_logic;
        br_o_op     : out std_logic
    );
end entity breathing;

architecture rtl of breathing is

    component pwmc is
    generic(
        p_dw : integer := 8
    );
    port(
        pwmc_i_clk    : in  std_logic;
        pwmc_i_rstn   : in  std_logic;
        pwmc_i_enable : in  std_logic;
        pwmc_i_offt   : in  std_logic_vector((p_dw-1) downto 0);
        pwmc_i_period : in  std_logic_vector((p_dw-1) downto 0);
        pwmc_o_pwmop  : out std_logic
    );
    end component pwmc;

    type grafect is (X0, X1, X2, X3, X4);
    signal w_next_state : grafect;
    signal r_curr_state : grafect;

    signal w_dcnt : std_logic_vector((dcnt_dw-1) downto 0);
    signal r_dcnt : std_logic_vector((dcnt_dw-1) downto 0);
    signal w_lg   : std_logic_vector((lg_dw-1) downto 0);
    signal r_lg   : std_logic_vector((lg_dw-1) downto 0);


begin

    gstate: process(br_i_clk, br_i_rstn)
    begin
        if br_i_rstn = '0' then
            r_curr_state <= X0;
        elsif rising_edge(br_i_clk) then
            r_curr_state <= w_next_state;
        end if;
    end process;

    control_path: process(r_curr_state, br_i_enable, r_dcnt, r_lg)
    begin
        case r_curr_state is
            when X0 =>
                if br_i_enable = '1' then
                    w_next_state <= X1;
                else
                    w_next_state <= X0;
                end if;
            when X1 =>
                if r_dcnt = X"FE" then -- (X"FFFE") set X"FE" for simulation
                    w_next_state <= X2;
                else
                    w_next_state <= X1;
                end if;
            when X2 =>
                if r_lg = X"F" then    -- (X"F0") set X"F" for simulation
                    w_next_state <= X3;
                else
                    w_next_state <= X1;
                end if;
            when X3 =>
                if r_dcnt = X"FE" then -- (X"FFFE") set X"FE" for simulation
                    w_next_state <= X4;
                else
                    w_next_state <= X3;
                end if;
            when X4 =>
                if r_lg = X"01" and br_i_enable = '1' then
                    w_next_state <= X1;
                elsif r_lg = X"01" and br_i_enable = '0' then
                    w_next_state <= X0;
                else
                    w_next_state <= X3;
                end if;
        end case;
    end process;

    data_path: process(r_curr_state, r_dcnt, r_lg)
    begin
        case r_curr_state is
            when X0 =>
                w_dcnt <= (others => '0');
                w_lg <= (others => '0');
            when X1 =>
                w_dcnt <= r_dcnt + '1';
                w_lg <= r_lg;
            when X2 =>
                w_dcnt <= (others => '0');
                w_lg <= r_lg + '1';
            when X3 =>
                w_dcnt <= r_dcnt + '1';
                w_lg <= r_lg;
            when X4 =>
                w_dcnt <= (others => '0');
                w_lg <= r_lg - '1';
        end case;
    end process;


    outbuf_path: process(br_i_clk, br_i_rstn)
    begin
        if br_i_rstn = '0' then
            r_dcnt <= (others => '0');
            r_lg <= (others => '0');
        elsif rising_edge(br_i_clk) then
            r_dcnt <= w_dcnt;
            r_lg <= w_lg;
        end if;
    end process;

    u0: pwmc
    generic map(
        p_dw => lg_dw
    )
    port map(
        pwmc_i_clk    => br_i_clk,
        pwmc_i_rstn   => br_i_rstn,
        pwmc_i_enable => br_i_enable,
        pwmc_i_offt   => r_lg,
        pwmc_i_period => X"FF",
        pwmc_o_pwmop  => br_o_op
    );

end rtl;
```

隨堂練習

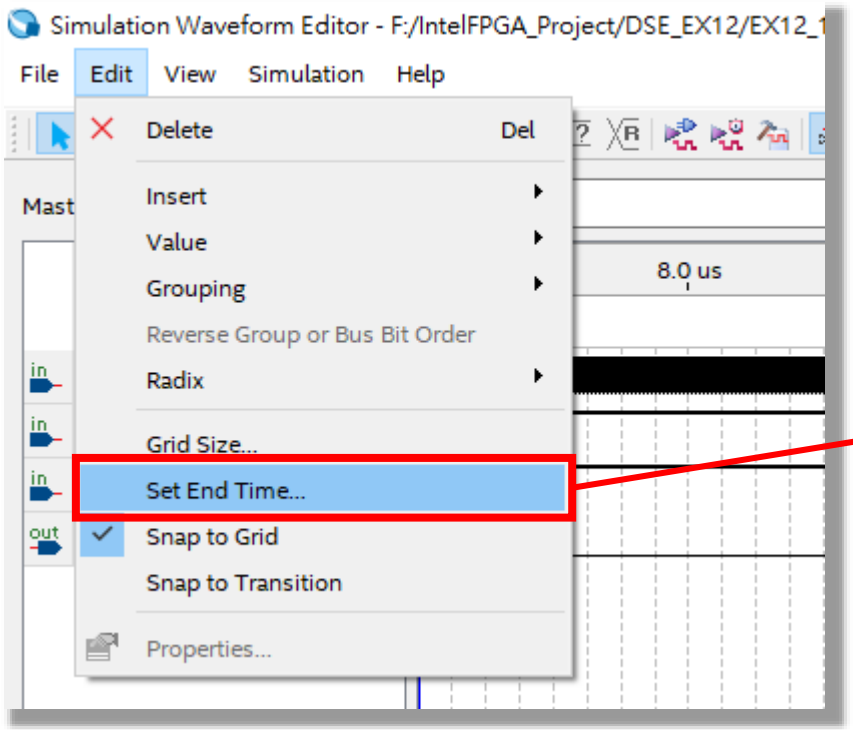
- 請使用VHDL完成上述PWM控制電路與呼吸燈控制電路，並完成紀錄，包括 **GRAFCET離散事件建模、VHDL Source Code、模擬波形圖，並將電路燒至實驗版中進行驗證**。

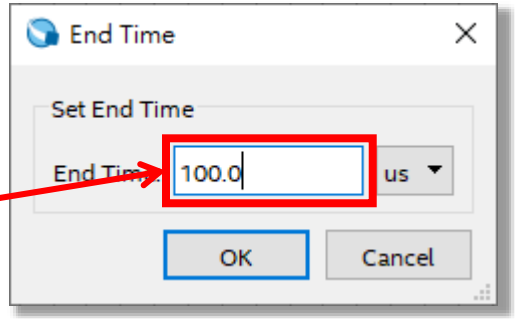
- 本次實驗完成後請，將專案與報告壓縮上傳EE-Class。
- Lecture12_組別XX.ZIP

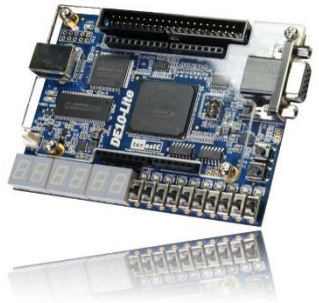


隨堂練習

- 模擬波形圖
  - 設定顯示時間

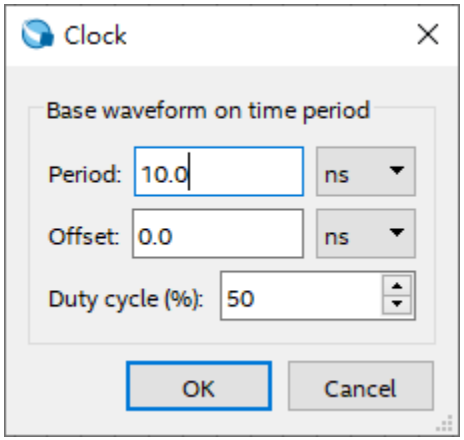


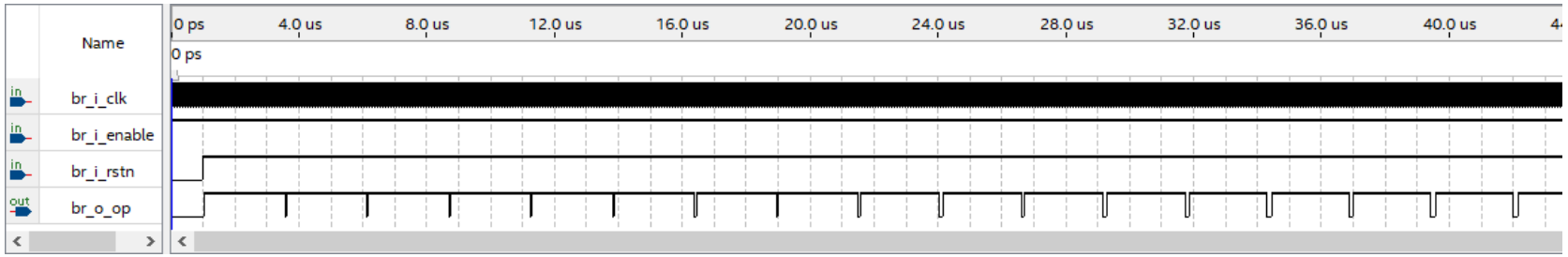


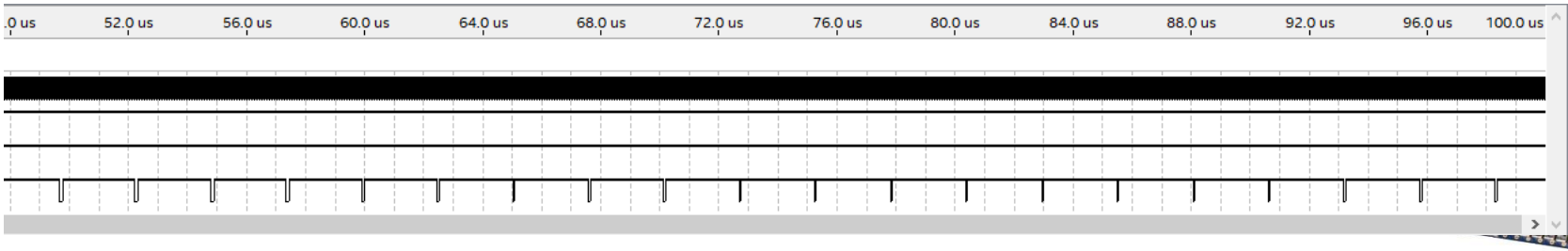


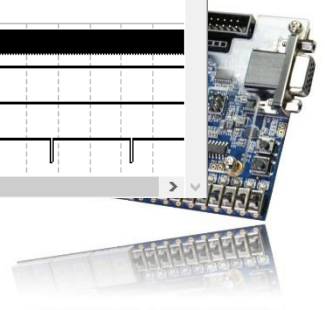
隨堂練習

- 模擬波形圖
  - 設定Clock

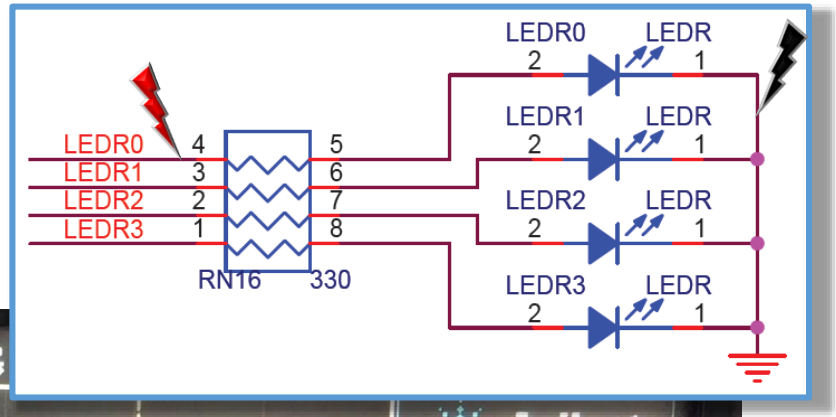


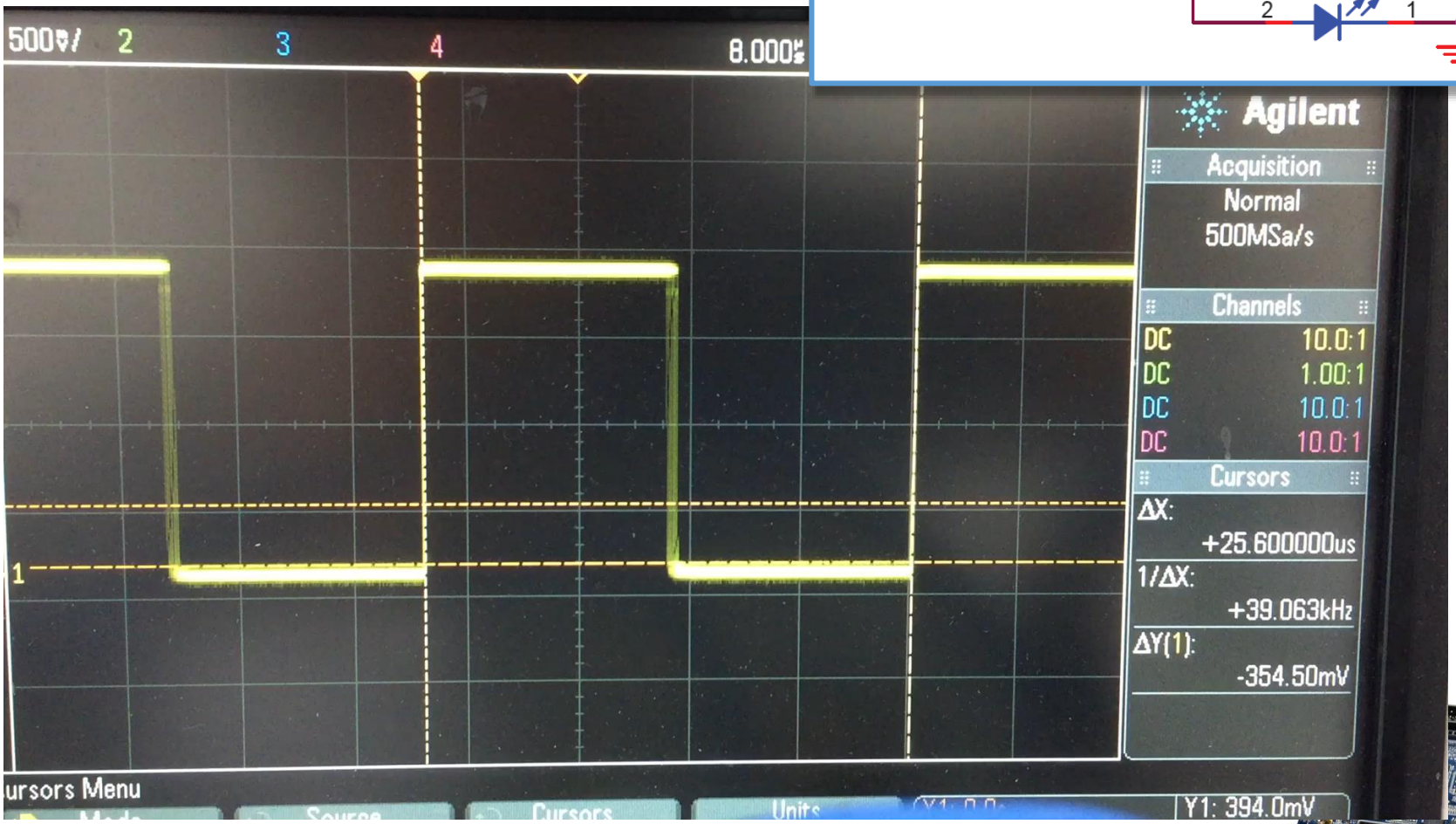




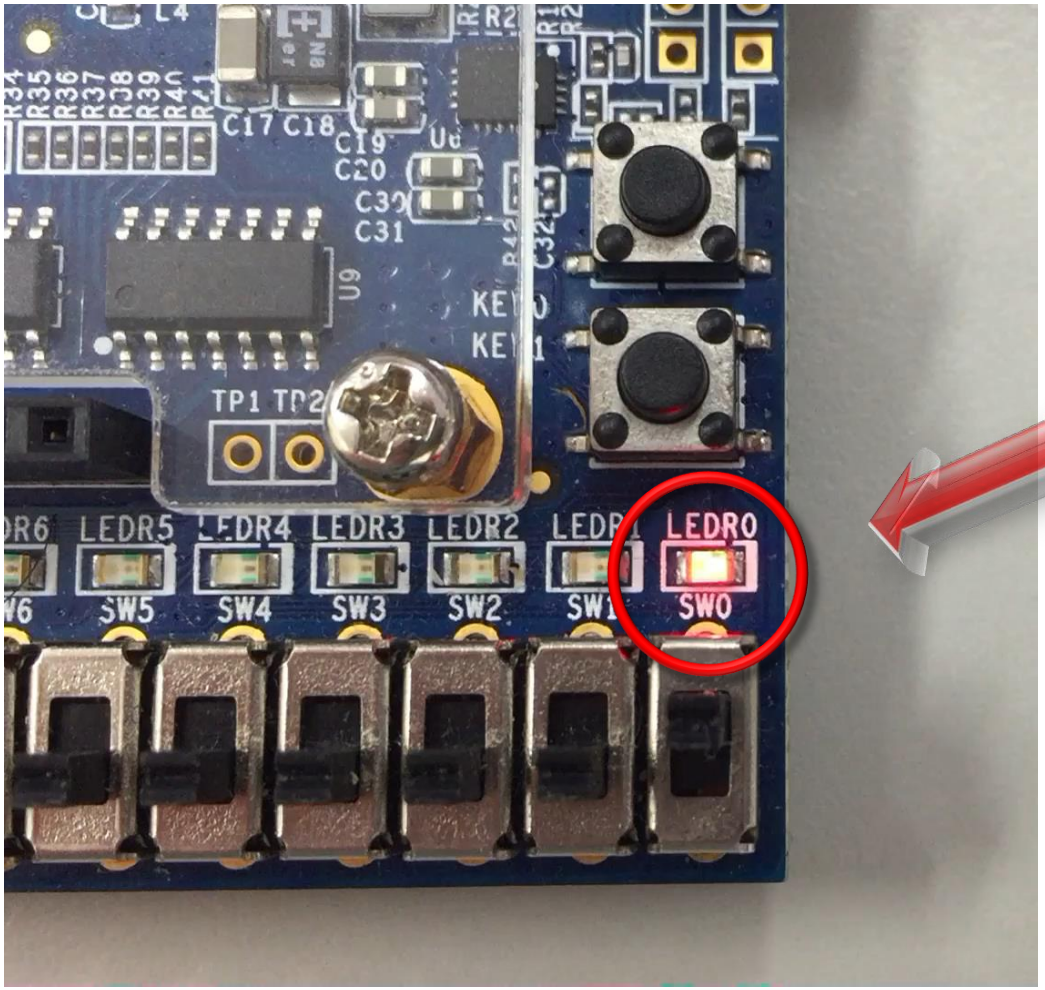


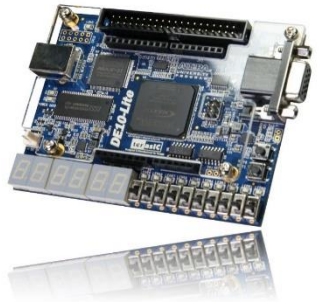
隨堂練習


LEDR0
LEDR1
LEDR2
LEDR3
4
3
2
1
5
6
7
8
RN16
330
LEDR0
LEDR1
LEDR2
LEDR3
LEDR
2
1


500mV
2
3
4
8.000us
Agilent
Acquisition
Normal
500MSa/s
Channels
DC 10.0:1
DC 1.00:1
DC 10.0:1
DC 10.0:1
Cursors
ΔX:
+25.600000us
1/ΔX:
+39.063kHz
ΔY(1):
-354.50mV
1
ursors Menu
Source
Cursors
Units
Y1: 394.0mV

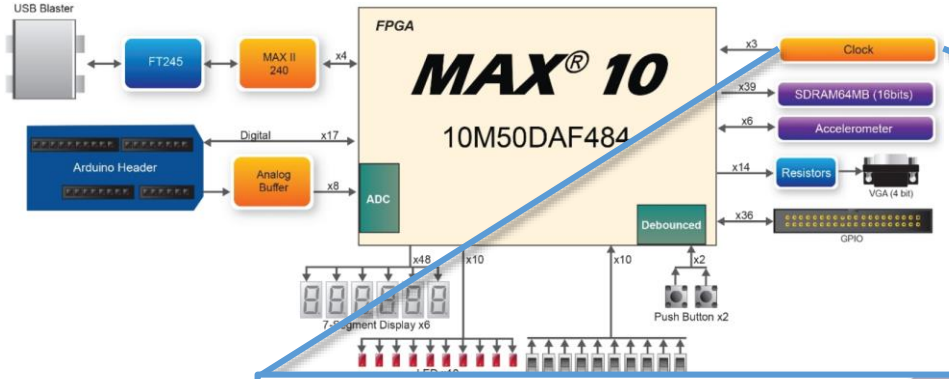
隨堂練習

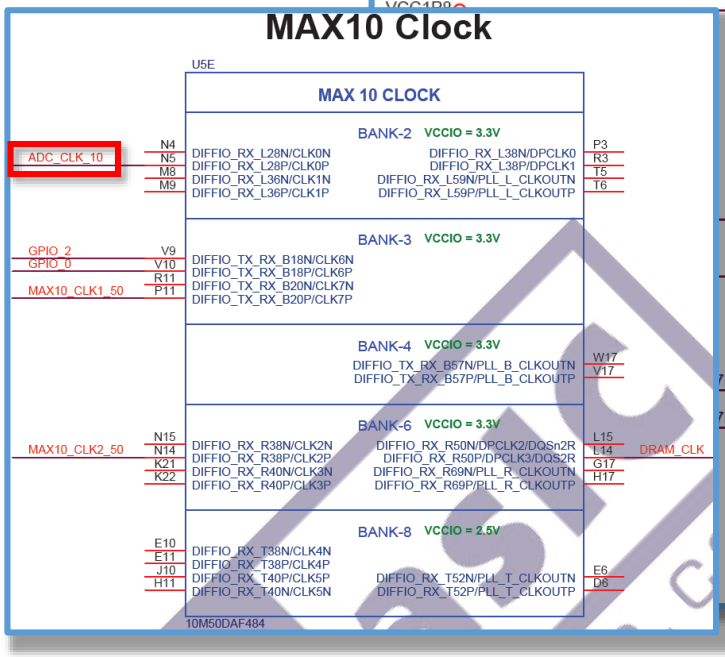


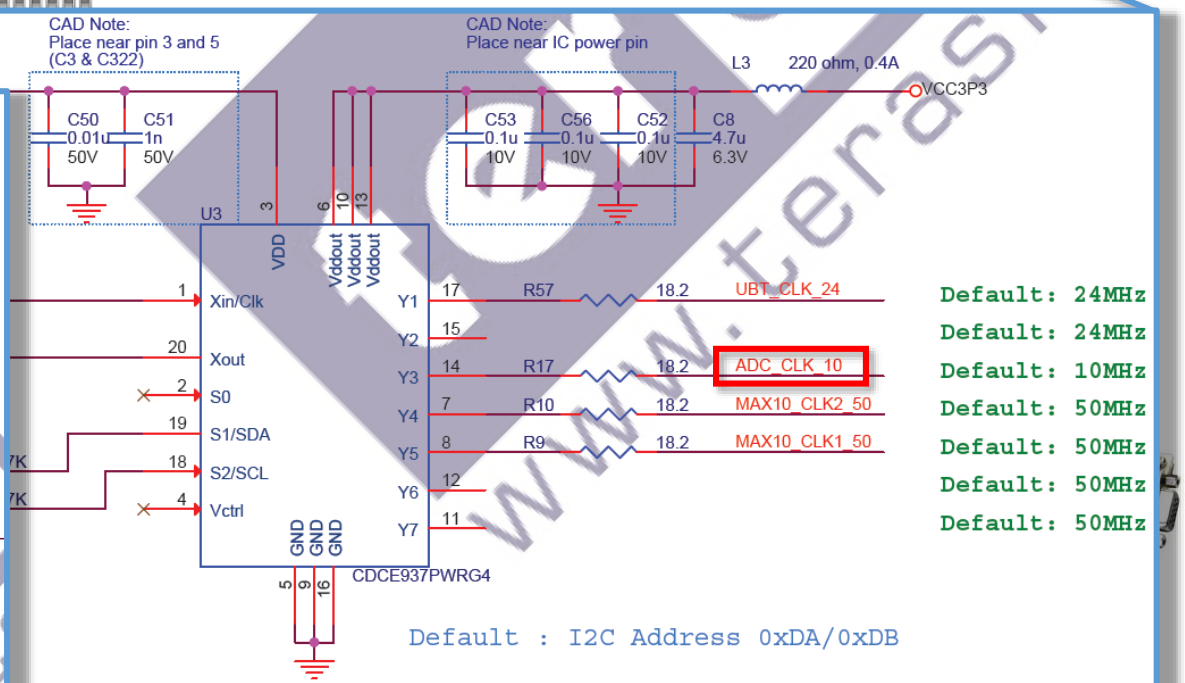


Clocks Circuit of DE10-Lite

(br_i_clk)

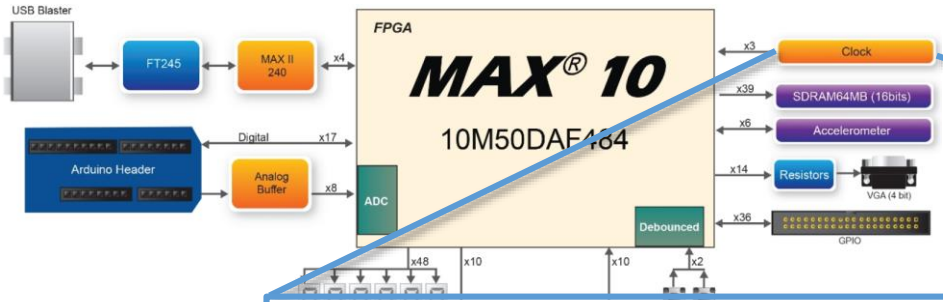


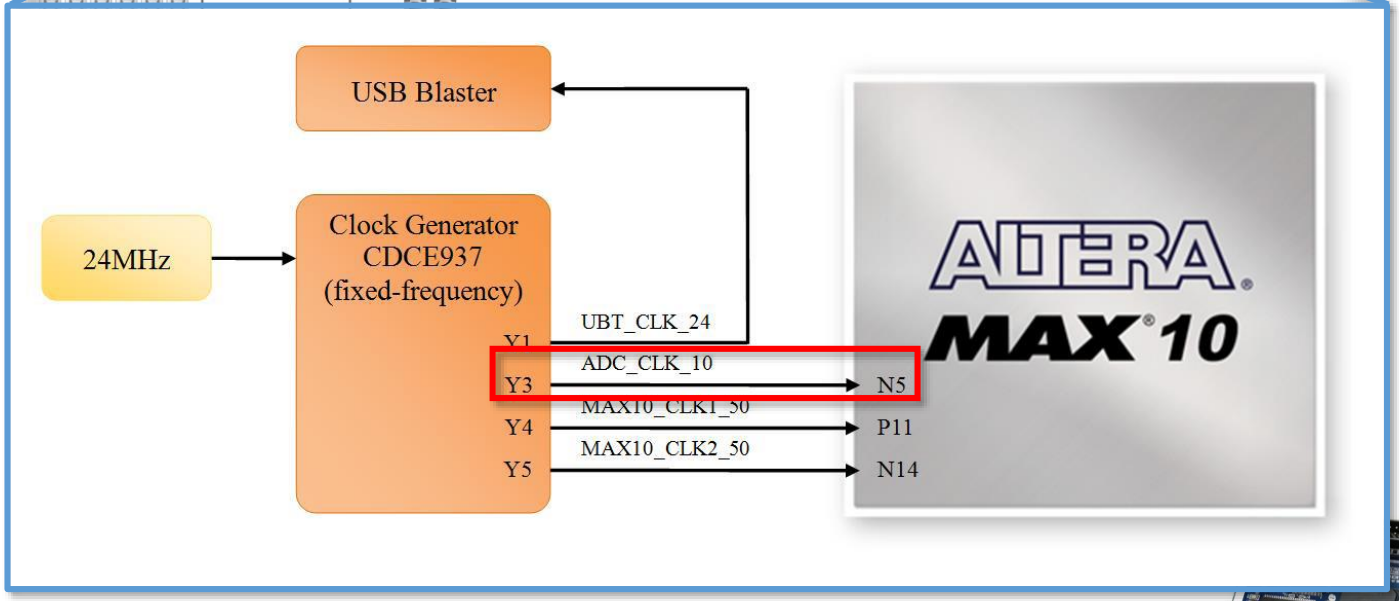




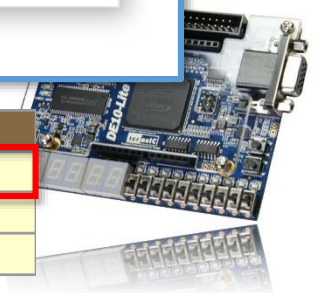
Clocks Circuit of DE10-Lite

(br_i_clk)



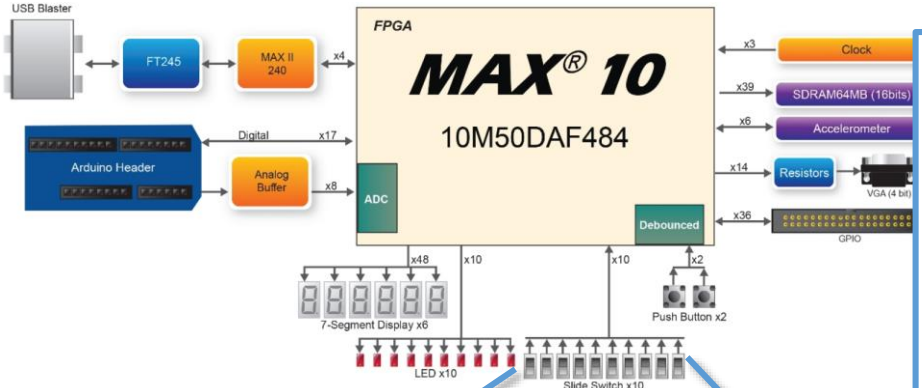


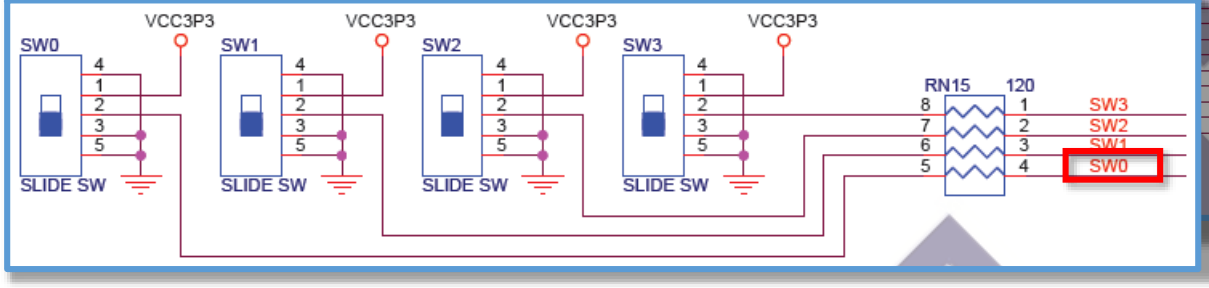
Signal Name	FPGA Pin No.	Description	I/O Standard
ADC_CLK_10	PIN_N5	10 MHz clock input for ADC (Bank 3B)	3.3-V LVTTL
MAX10_CLK1_50	PIN_P11	50 MHz clock input(Bank 3B)	3.3-V LVTTL
MAX10_CLK2_50	PIN_N14	50 MHz clock input(Bank 3B)	3.3-V LVTTL



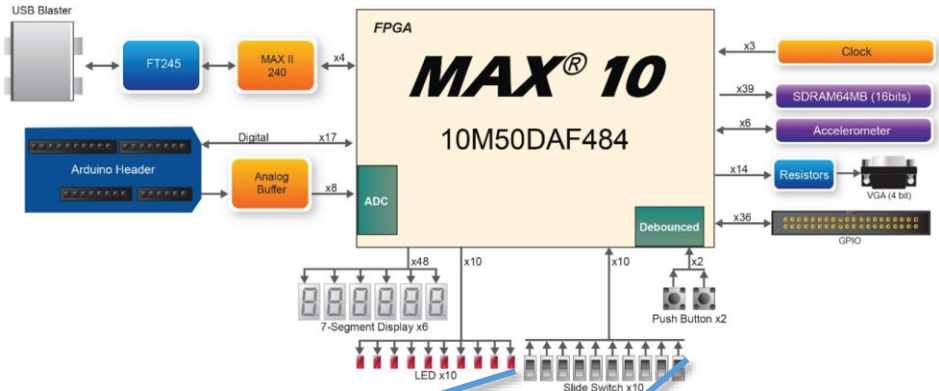
Slide Switches Circuit of DE10-Lite

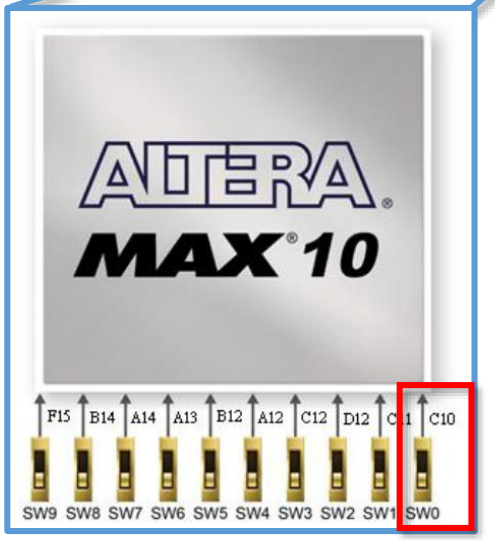
(br_i_enable)





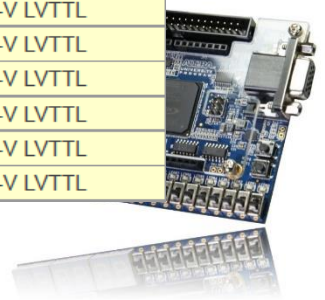
Slide Switches Circuit of DE10-Lite





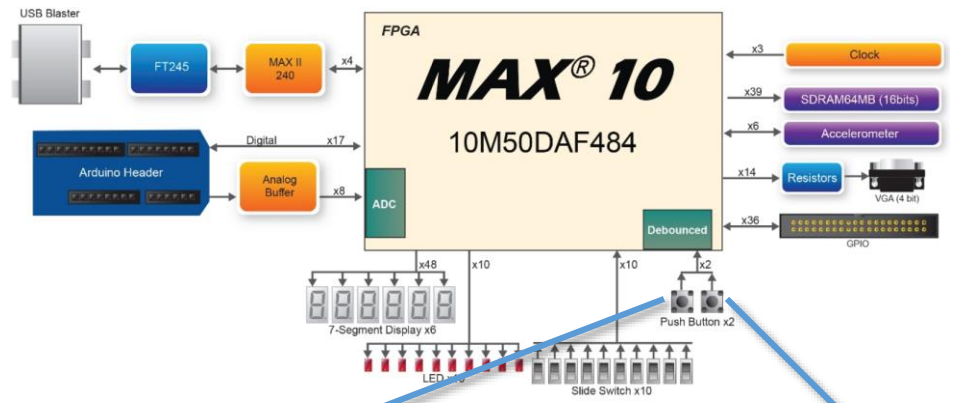
{br_i_enable}

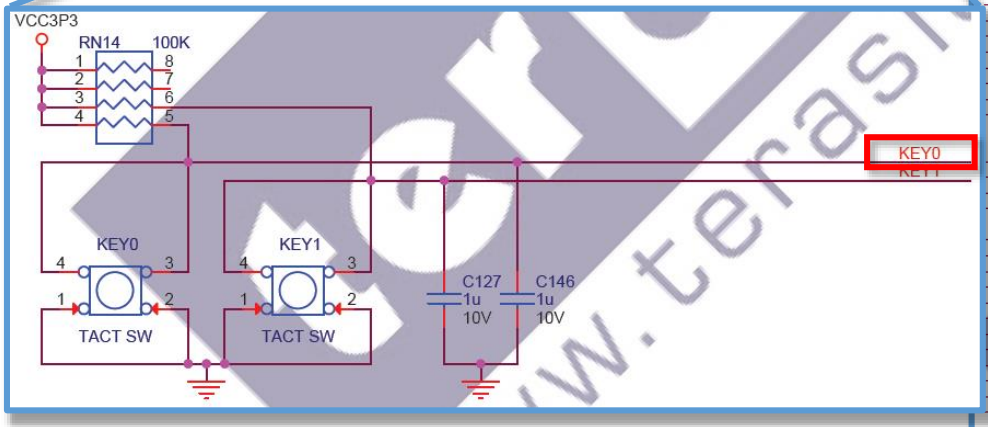
Signal Name	FPGA Pin No.	Description	I/O Standard
SW0	PIN_C10	Slide Switch[0]	3.3-V LVTTL
SW1	PIN_C11	Slide Switch[1]	3.3-V LVTTL
SW2	PIN_D12	Slide Switch[2]	3.3-V LVTTL
SW3	PIN_C12	Slide Switch[3]	3.3-V LVTTL
SW4	PIN_A12	Slide Switch[4]	3.3-V LVTTL
SW5	PIN_B12	Slide Switch[5]	3.3-V LVTTL
SW6	PIN_A13	Slide Switch[6]	3.3-V LVTTL
SW7	PIN_A14	Slide Switch[7]	3.3-V LVTTL
SW8	PIN_B14	Slide Switch[8]	3.3-V LVTTL
SW9	PIN_F15	Slide Switch[9]	3.3-V LVTTL

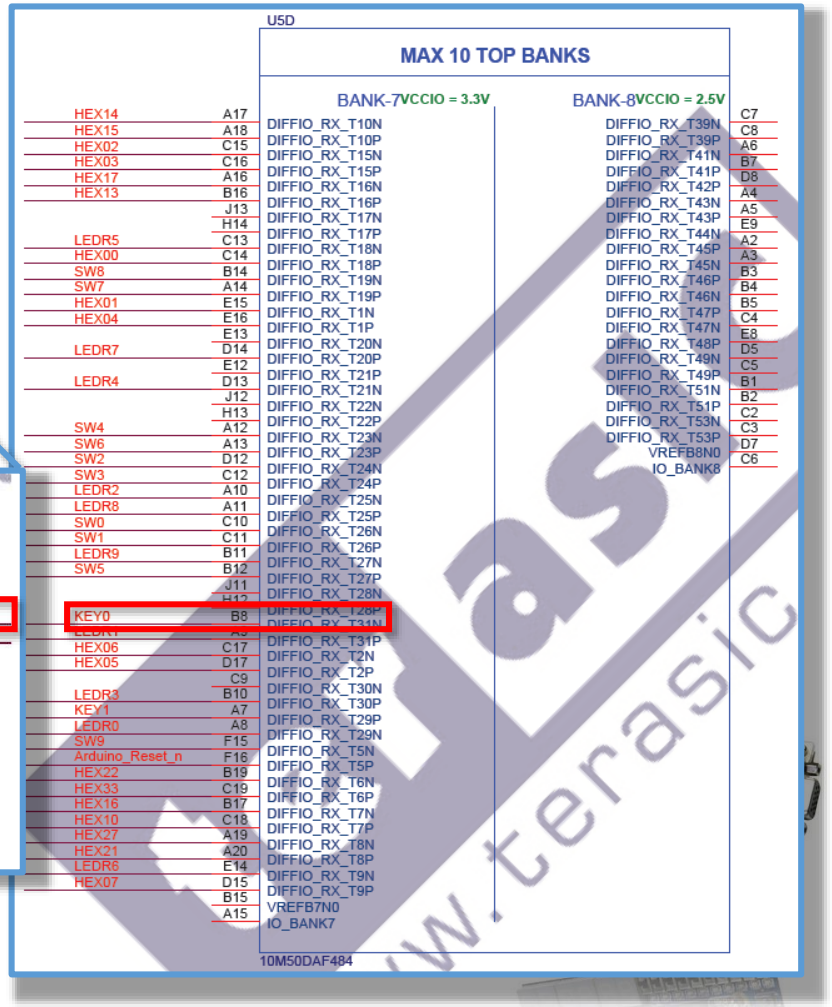


Push Buttons Circuit of DE10-Lite

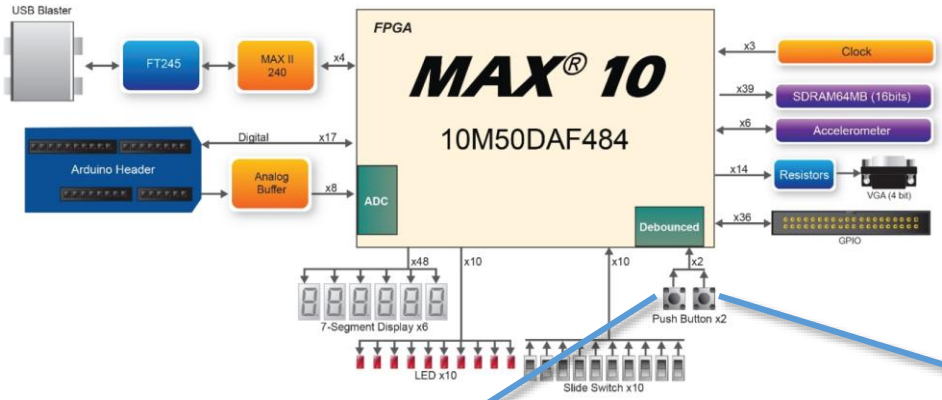
(br_i_rstn)

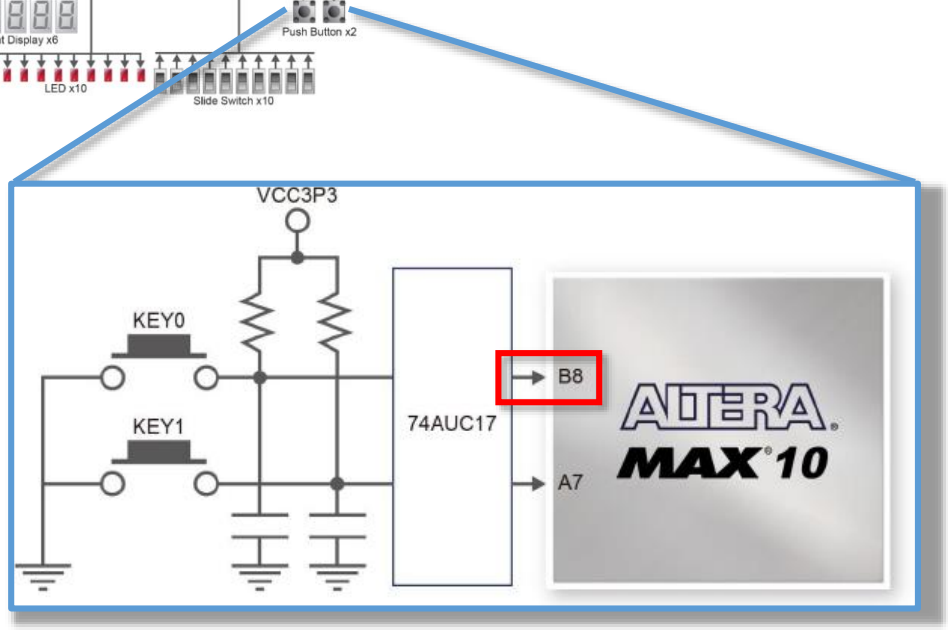






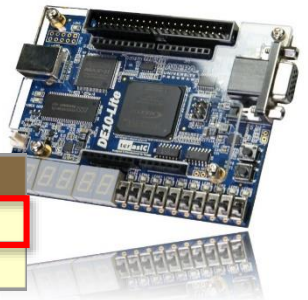
Push Buttons Circuit of DE10-Lite





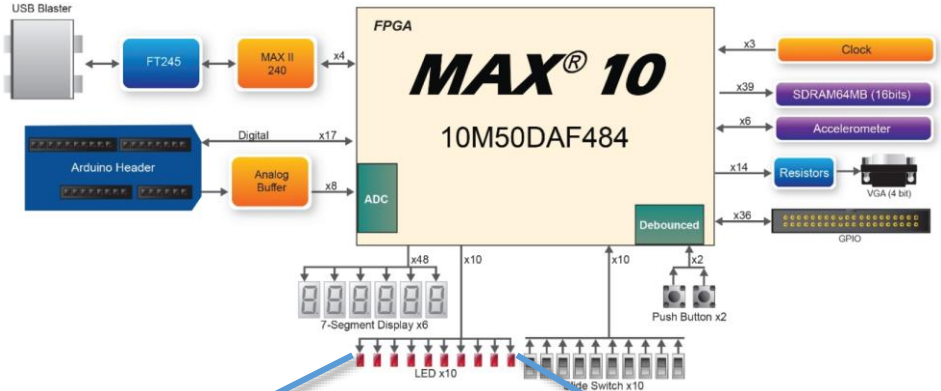
(br_i_rstn)

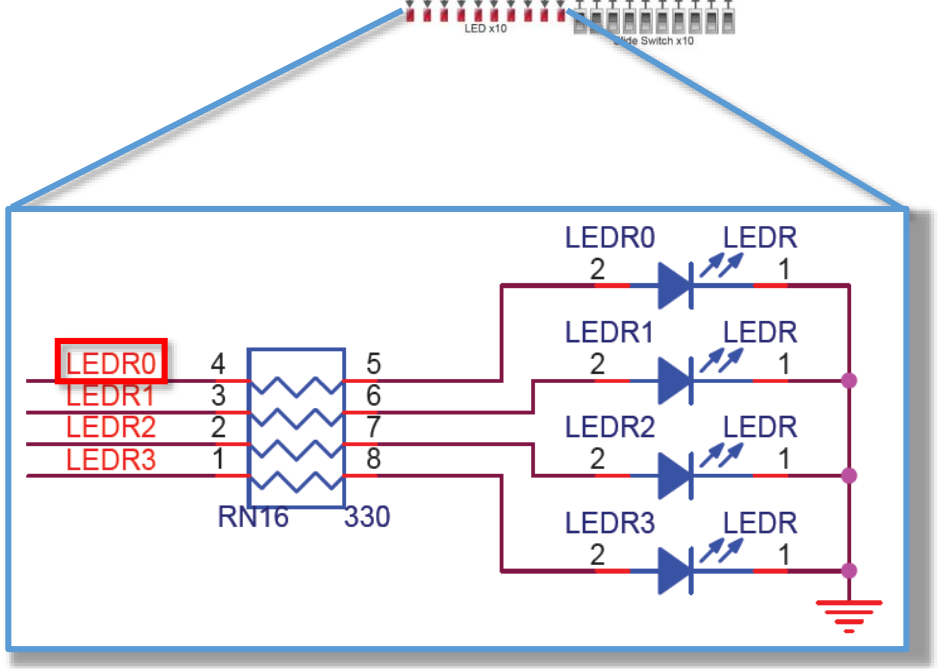
Signal Name	FPGA Pin No.	Description	I/O Standard
KEY0	PIN_B8	Push-button[0]	3.3 V SCHMITT TRIGGER"
KEY1	PIN_A7	Push-button[1]	3.3 V SCHMITT TRIGGER"



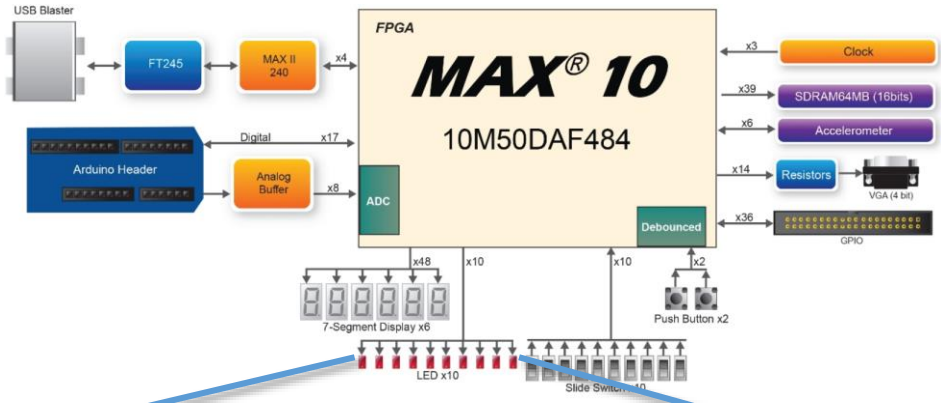
LEDs Circuit of DE10-Lite

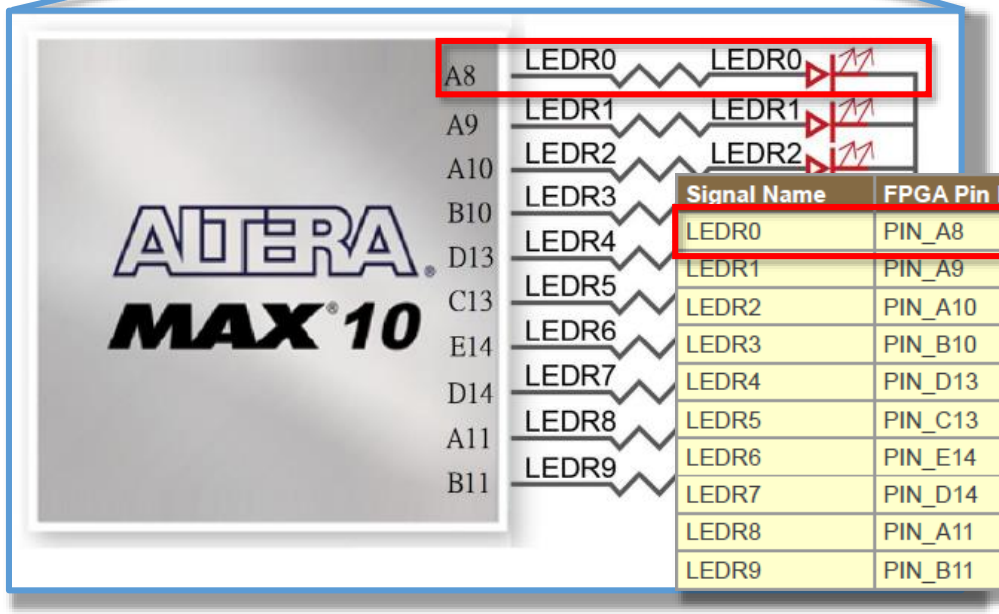
[br_o_op]





LEDs Circuit of DE10-Lite





Signal Name	FPGA Pin No.	Description	I/O Standard
LEDR0	PIN_A8	LED [0]	3.3-V LVTTL
LEDR1	PIN_A9	LED [1]	3.3-V LVTTL
LEDR2	PIN_A10	LED [2]	3.3-V LVTTL
LEDR3	PIN_B10	LED [3]	3.3-V LVTTL
LEDR4	PIN_D13	LED [4]	3.3-V LVTTL
LEDR5	PIN_C13	LED [5]	3.3-V LVTTL
LEDR6	PIN_E14	LED [6]	3.3-V LVTTL
LEDR7	PIN_D14	LED [7]	3.3-V LVTTL
LEDR8	PIN_A11	LED [8]	3.3-V LVTTL
LEDR9	PIN_B11	LED [9]	3.3-V LVTTL

[br_o_op]