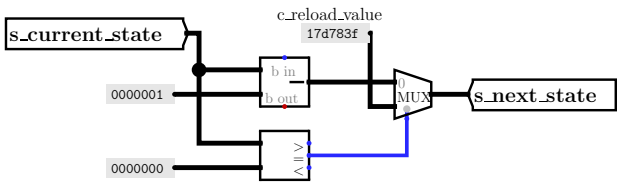
Exercise 1:

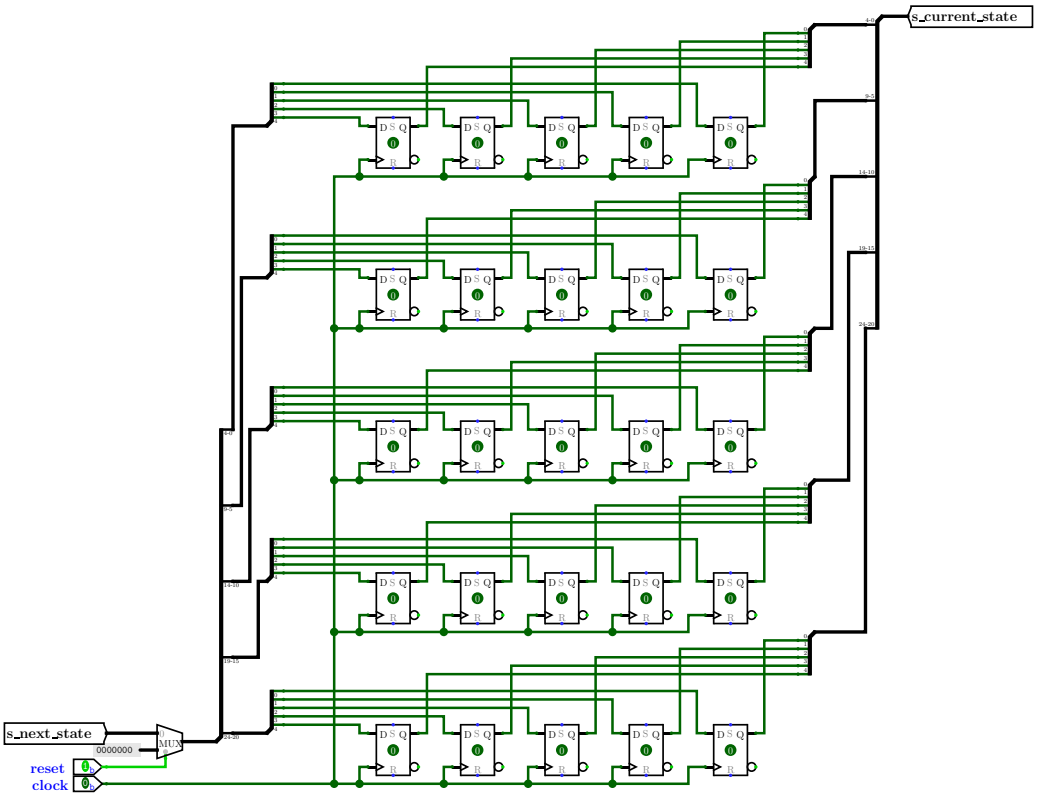
Review the file `divider-behavior.vhdl` and try to understand its functionality. Answer all the questions asked in this file.

```
ARCHITECTURE no_target_specific OF clock_divider IS

   CONSTANT c_reload_value : unsigned( 24 DOWNTO 0 ) :=
              to_unsigned(24999999,25);
   -- The above constant represents (25000000 - 1) which corresponds to
   -- the divider value required for a 50 MHz clock to generate an enable pulse
   -- at a frequency of 2 Hz. We have to deduct 1 above, as "0" is also a valid
   -- value, and the interpretation used for this constant is binairy.
   SIGNAL s_current_state  : unsigned( 24 DOWNTO 0 );
   SIGNAL s_next_state     : unsigned( 24 DOWNTO 0 );

BEGIN

   -------------------------------------------------------------------------------
   --- Here the update logic of the FSM of type Moore is described:          ---
   -------------------------------------------------------------------------------
   s_next_state <= c_reload_value WHEN s_current_state = to_unsigned(0,25) ELSE
                   s_current_state - 1;
   -- Question1: What is the hardware that is described in the above implicit
   --            process?

   -------------------------------------------------------------------------------
   --- Here the state logic of the FSM of type Moore is described:           ---
   -------------------------------------------------------------------------------
   make_state : PROCESS( clock , reset , s_next_state )
   BEGIN
      IF (rising_edge(clock)) THEN
         IF (reset = '1') THEN s_current_state <= (OTHERS => '0');
                          ELSE s_current_state <= s_next_state;
         END IF;
      END IF;
   END PROCESS make_state;
   -- Question2: What is the hardware that is described in the above explicit
   --            process, and is the reset synchronous or asynchronous?

   -------------------------------------------------------------------------------
   --- Here the output logic of the FSM of type Moore is described:          ---
   -------------------------------------------------------------------------------
   enable <= '1' WHEN s_current_state = to_unsigned(1,25) ELSE '0';
   -- Question3: What is the hardware that is described in the above implicit
   --            process?

END no_target_specific;
```

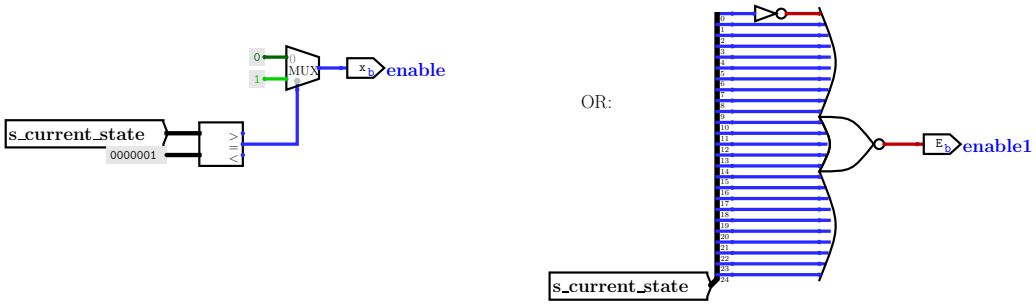
Answers to the questions above:
Question 1:



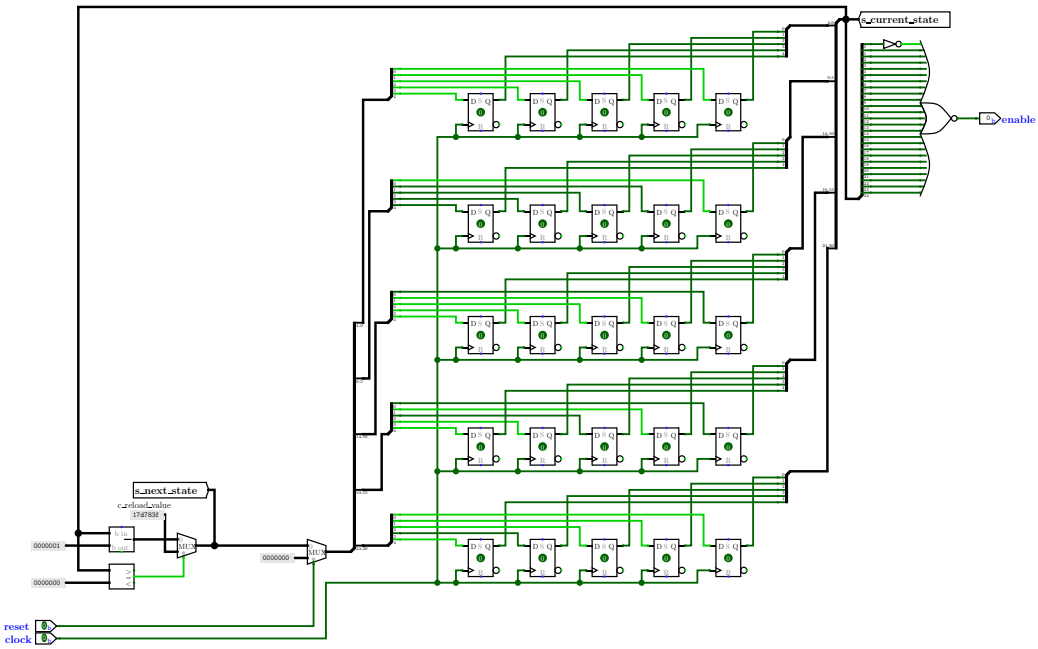
Question 2:
We have a synchronous reset and the circuit described is:



Question 3:

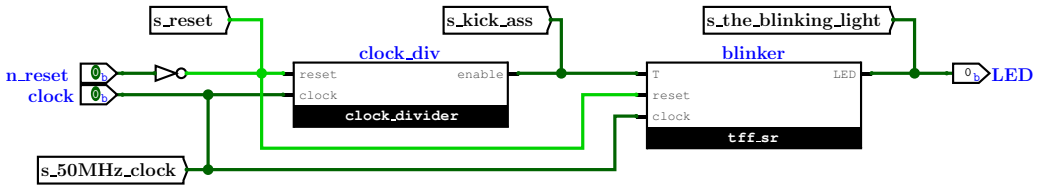


The whole system described is than:



Exercise 2:

Review the file `laboratory6_top-behavior.vhdl` and draw the block diagram of the entire system.



Exercise 3:

Proceed with the same procedure as for the clock and buttons for the **LED** pins. You can use an LED of the LED-array or one segment out of a seven segment display. You can find the information needed in the `leds` tab in your web browser.

IMPORTANT:

The TCL scripts are case sensitive. This means you need to use the same uppercase or lowercase letters for the signal names in the TCL script as you did in the entity.

IMPORTANT:

All inputs and outputs need to be connected to a pin. If not, Quartus will automatically assign the inputs and outputs to a pin. Since Quartus doesn't know about which pin is connected to which element on the Board it can happen that two outputs are connected together. This causes short cuts and finally the holy smoke.

Exercise 4:

Modify the parameter of the clock such as the clock is connected to the 12 MHz clock source. Regenerate the bit file and download it to the GECKO4Education. What do you expect to observe?

The lamp will blink a factor of $\frac{50}{12} \approx 4.2$ times slower.

IMPORTANT:

It is important that you understand and that you can fully reproduce all the explained steps in this exercise. In the next exercises and in the following semester we expect you to use Modelsim and Quartus on your own.

Exercise 5:

Draw the output signal of the counter qualitatively. How many clock periods will the signal be 1 and how many periods will it be 0?

The reasoning for this drawing:
When we look at the transition logic of the counter (Question 1 in the vhdl-description) we can observe that the next state equals to the current state - 1 if the binairy value of current state not equals to zero. In the case that current state equals to zero than the next state is assigned the binary value 24999999. This means that:

1. Our state machine has exactly 25000000 states.

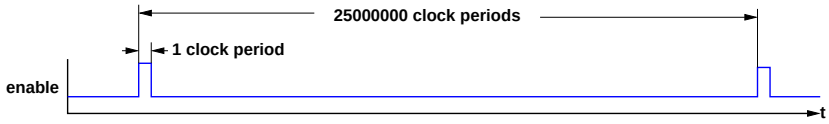
2. Our state machine stays exactly 1 clock period in a given state.

The output signal (enable) is one in case the state machine (current state) equals to the binary value 1. In all other cases it is zero (see Question 3 in the vhdl-description). Hence combining this information gives us:

1. The enable signal is exactly 1 clock period one.

2. The enable signal is exactly (25000000-1)=24999999 clock periods at zero.

3. This sequence repeats itself each 25000000 clock periods.



Exercise 6:

Simulate the entire system in Modelsim.

Exercise 7:

Modify the VHDL description such as the blinking light blinks with a frequency of 10 Hz using the 12 MHz clock.

So lets analyze this problem:

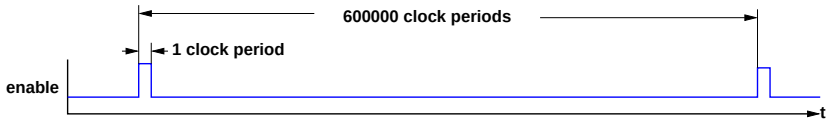
1. The lamp has to blink with 10 Hz, this means that the tff_sr state machine has to change state 20 times in a second (10 times from 0 to 1 and 10 times from 1 to 0).

2. The tff_sr changes state each time that the T-input is one, the reset input is zero and we have a rising edge of the clock (see TP5).

3. To have the 20 changed per second we have to generate a T-signal that is active during 1 period of the clock for exactly 20 times per second (see also the answer to exercise 5).

4. As we have a clock frequency of 12MHz, we have to have a repetition rate of the enable signal of $\frac{12000000}{20} = 600000$ clock periods.

5. Hence to implement this functionality we have to generate following signal with the counter:



Thus we require a counter with 600000 states instead of 25000000. To represent 600000 states we require $\left\lceil \frac{ln(600000)}{ln(2)} \right\rceil = 20$ bits. The new VHDL-description is than (note that we only have to modify the constant value and the number of bits used):

```
ARCHITECTURE new_implementation OF clock_divider IS

   CONSTANT c_reload_value : unsigned( 19 DOWNTO 0 ) :=
               to_unsigned(599999,25);
   -- The above constant represents (600000 - 1) which corresponds to
   -- the divider value required for a 12 MHz clock to generate an enable pulse
   -- at a frequency of 20 Hz. We have to deduct 1 above, as "0" is also a valid
   -- value, and the interpretation used for this constant is binairy.
   SIGNAL s_current_state  : unsigned( 19 DOWNTO 0 );
   SIGNAL s_next_state     : unsigned( 19 DOWNTO 0 );

BEGIN

   -------------------------------------------------------------------------
   --- Here the update logic of the FSM of type Moore is described:      ---
   -------------------------------------------------------------------------
   s_next_state <= c_reload_value WHEN s_current_state = to_unsigned(0,20) ELSE
                   s_current_state - 1;

   -------------------------------------------------------------------------
   --- Here the state logic of the FSM of type Moore is described:       ---
   -------------------------------------------------------------------------
   make_state : PROCESS( clock , reset , s_next_state )
   BEGIN
      IF (rising_edge(clock)) THEN
         IF (reset = '1') THEN s_current_state <= (OTHERS => '0');
                          ELSE s_current_state <= s_next_state;
         END IF;
      END IF;
   END PROCESS make_state;

   -------------------------------------------------------------------------
   --- Here the output logic of the FSM of type Moore is described:      ---
   -------------------------------------------------------------------------
   enable <= '1' WHEN s_current_state = to_unsigned(1,20) ELSE '0';

END new_implementation;
```