Laboratory practice 5

ASM: Slot Machine

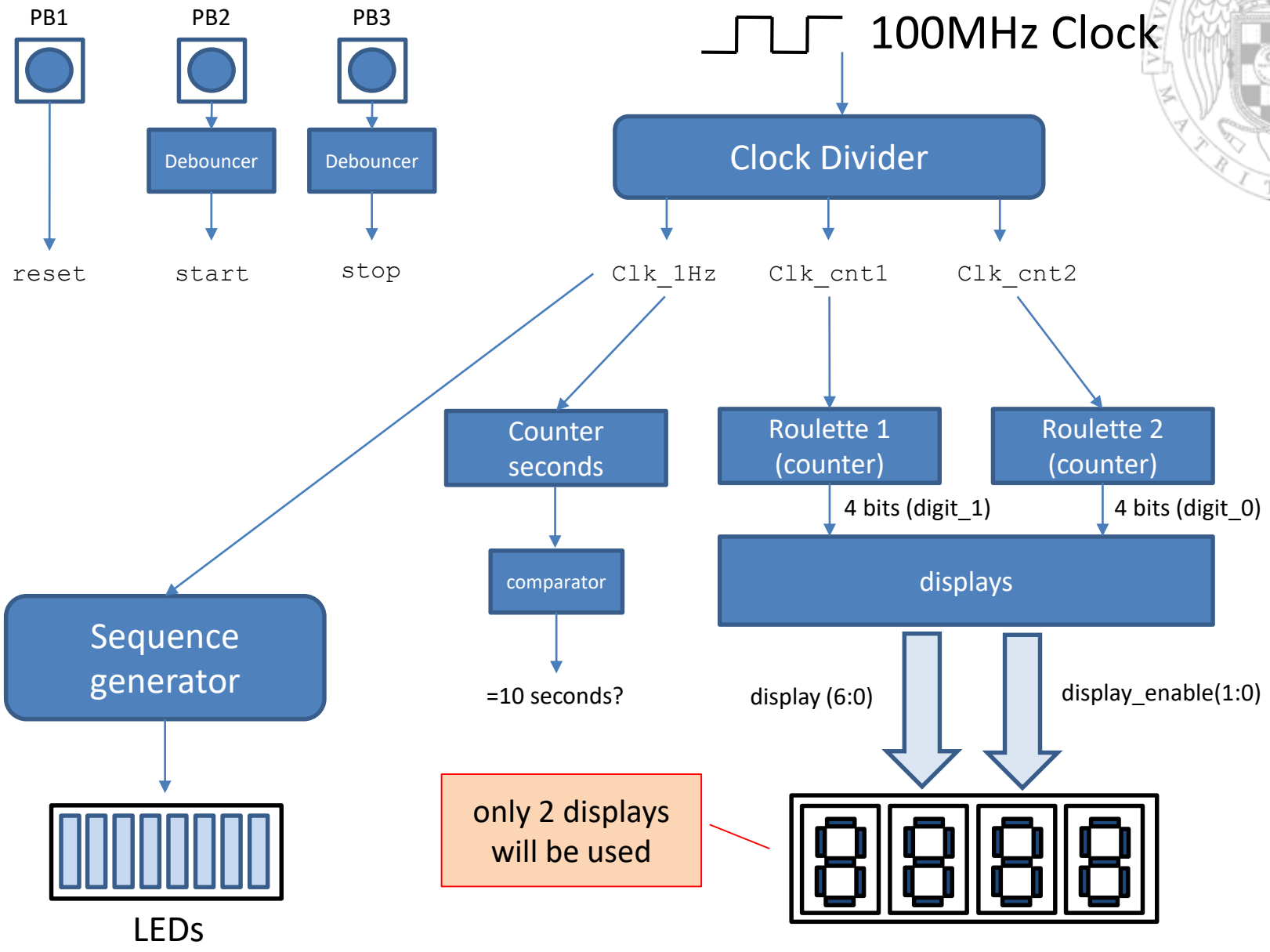
Specifications

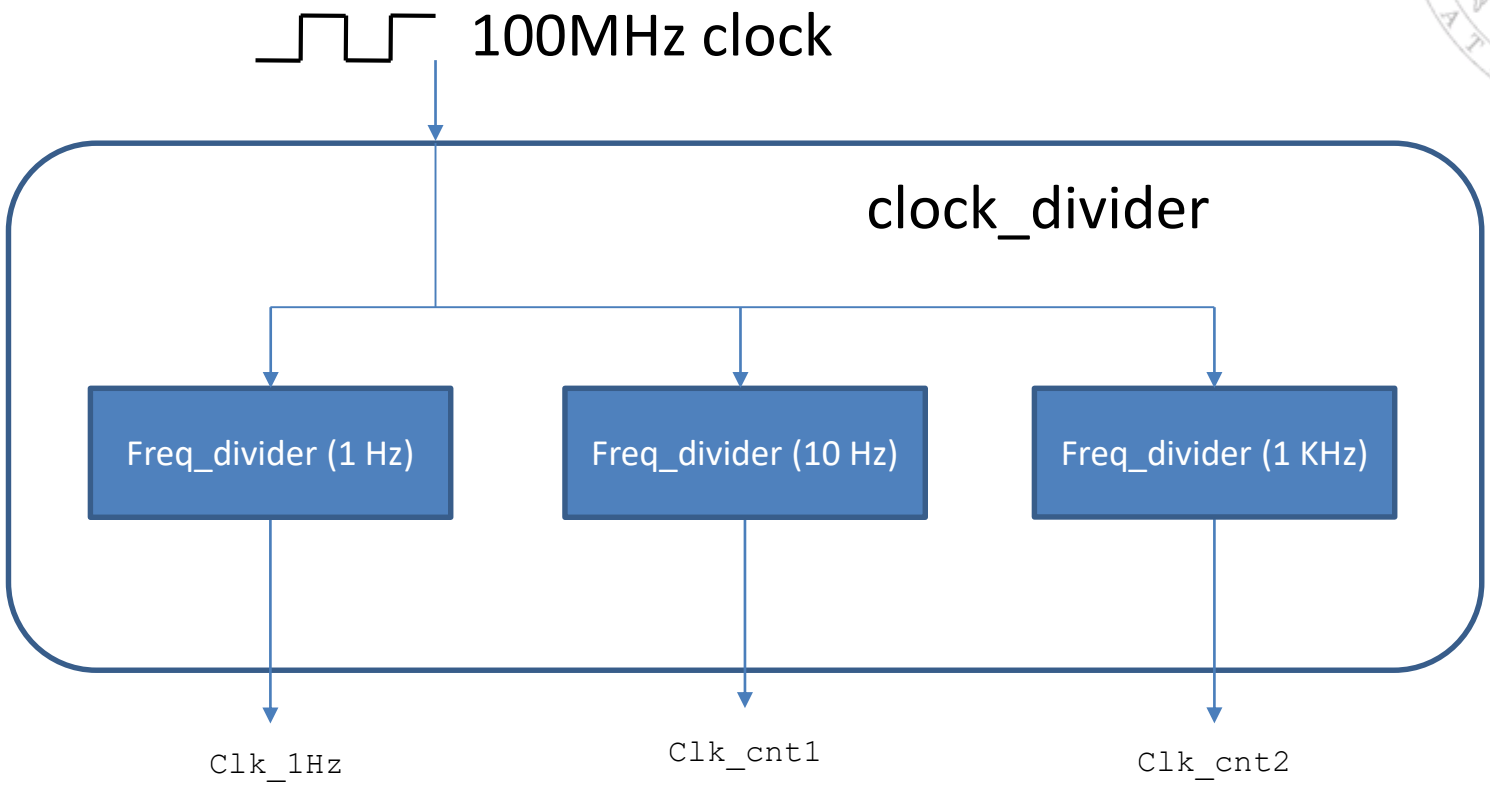
- The slot machine has two roulettes that we will implement using 2 mod-10 counters.
- Initially, the machine will try to attract customers by using the LEDs following a certain sequence.
- When the user presses '`start`', the counters will start counting, each one at a different frequency, and fast enough so that the numbers are not visible to the human eye.
- When the user presses '`stop`', the counters will stop. If the numbers of the counters are equals, the user wins; he/she loses otherwise.
- Depending on the result, the LEDs will perform a certain sequence for 10 seconds.
- After that, the machine returns to the initial state.

Elements

- Inputs:
  - 3 push buttons (`reset`, `start` and `stop`).
- Outputs:
  - 2 7-segment displays.
  - 10 LEDs.
- Control Unit/Data Path:
  - Sequence generator.
  - Frequency divider.
  - Clock divider.
  - Definitions (package). It should be completed.
  - Rebound filter.
  - Counters.
  - …

You can find these files in the CV

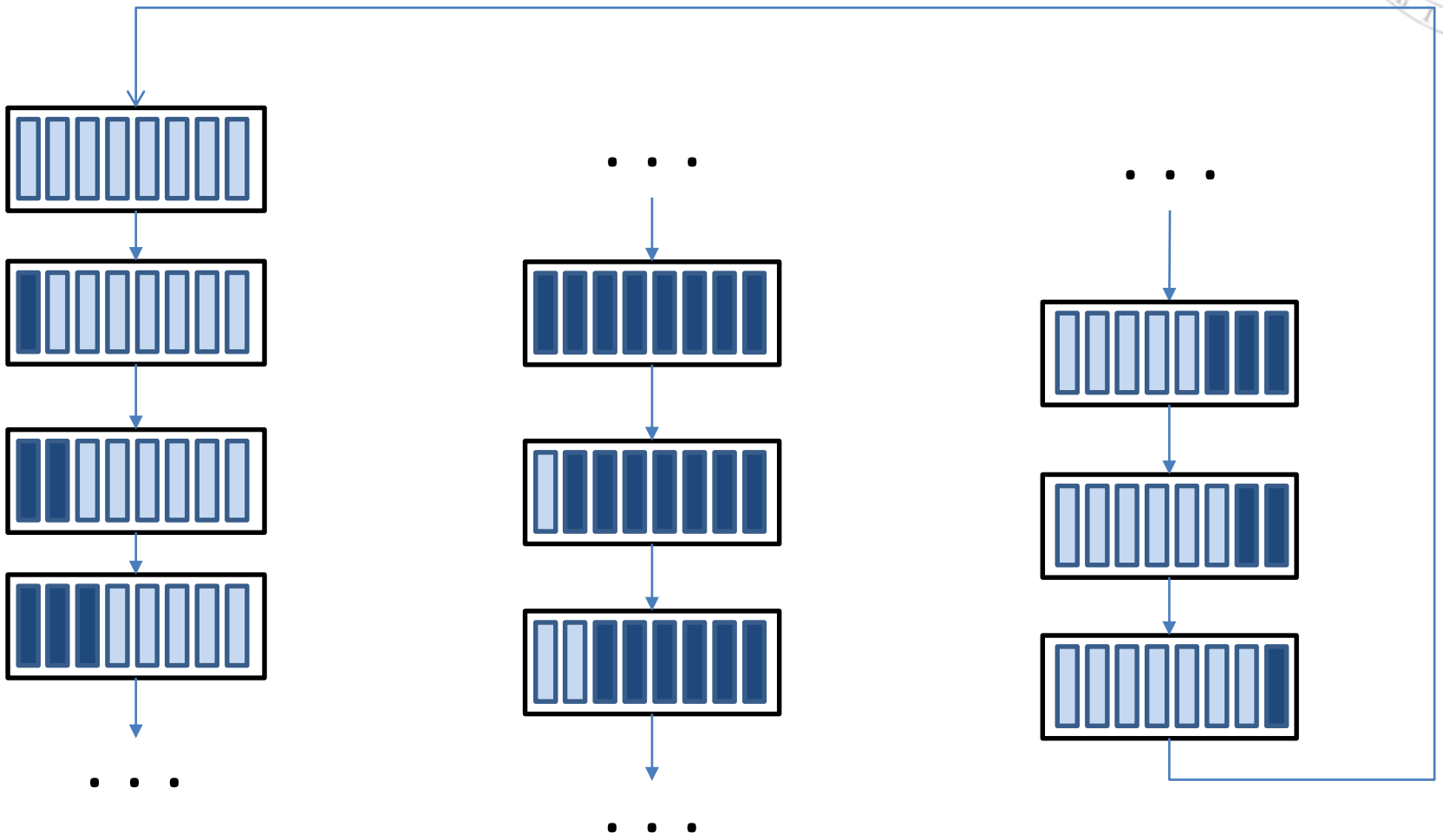

PB1
PB2
PB3
100MHz Clock
Debouncer
Debouncer
Clock Divider
reset
start
stop
Clk_1Hz
Clk_cnt1
Clk_cnt2
Counter seconds
Roulette 1 (counter)
Roulette 2 (counter)
4 bits (digit_1)
4 bits (digit_0)
comparator
displays
Sequence generator
=10 seconds?
display (6:0)
display_enable(1:0)
only 2 displays will be used
LEDs


100MHz clock
clock_divider
Freq_divider (1 Hz)
Freq_divider (10 Hz)
Freq_divider (1 KHz)
Clk_1Hz
Clk_cnt1
Clk_cnt2

Operating scheme

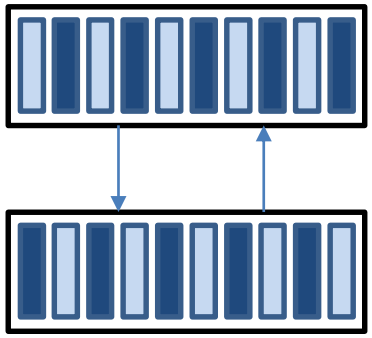
- Initial State (`reset`):
  - Reset counters.
  - Displays show “00”.
  - LEDs sequence: “attract customers”.
- Counting (after `start`):
  - LEDs off.
  - Counters counting at different speed.
  - Displays showing the count, which changes fast enough so that the numbers are not visible to the human eye.
- Result (after `stop`):
  - Roulettes stopped.
  - Displays showing the numbers of the roulettes.
  - LEDs sequence (for 10 seconds):
    - WINNER
    - LOSER

Sequence 1: attract customers

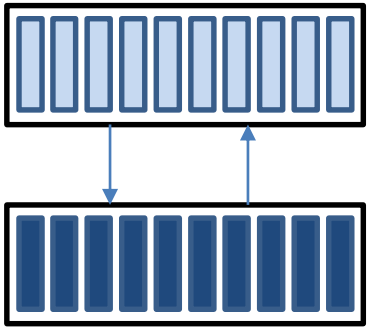


Sequences 2 and 3

- WINNER:



- LOSER:



Sequence generator

```
entity seq_generator is
    port (
        clk      : in std_logic;
        rst      : in std_logic;
        load     : in std_logic;
        rotate_invert : in std_logic;
        seq_in   : in  std_logic_vector(7 downto 0);
        seq_out  : out std_logic_vector(7 downto 0)
    );
end seq_generator;

architecture divider_arch of seq_generator is
    signal reg : std_logic_vector(7 downto 0);
begin

    counter:
    process(rst, clk, load, seq_in)
    begin
        if(rst='1') then
            reg <= (others=>'0');
        elsif rising_edge(clk) then
            if(load='1') then
                reg <= seq_in;
            elsif (rotate_invert = '1') then
                reg <= not(reg(0)) & reg(7 downto 1);
            else
                reg <= not(reg);
            end if;
        end if;
    end process counter;

    seq_out <= reg;

end divider_arch;
```

We can load (in parallel) an initial sequence (seq_in)

We can either rotate to the right (rotate_invert = '1') the sequence or invert it (rotate_invert = '0')

The sequence is internally stored

When rotating (to the right), the inverted LSB becomes the new MSB

- 3 possible sequences:
  - Attract customers:
    - Initially load "00000000" or "11111111"
    - Then, rotate
  - WINNER:
    - Initially load "10101010"
    - Then, invert
  - LOSER:
    - Initially load "00000000"
    - Then, invert

Grading

- Slot Machine implemented homework
  - You can simulate it by creating your own test-bench with examples of winning and losing.
- The students must come to the laboratory with the **ASM and Data Path drawn** indicating clearly:
  - The control signals for each state
  - The state and intermediate signals
- The student must show the slot machine working, and has to understand the implementation and functionality (0.15 pts)
- Advanced part +0.35 pts
- Lab 5 is **NOT** recoverable