

PRE-LAB

- Recall that a frequency divider was provided in lab 5 to generate a 1-Hz clock signal (*CLOCK_1*) from a 50-MHz clock source (*CLOCK_50*) on a DE2 board (see Figure 1). The timing diagram for the *CLOCK_1* signal is shown in Figure 2. Using this frequency divider as a building block, draw a circuit that generates a 0.5 Hz clock signal (*CLOCK_05*) with 50% duty cycle (Hint: use one additional flip flop). The timing diagram for this *CLOCK_05* signal is also shown in Figure 2.

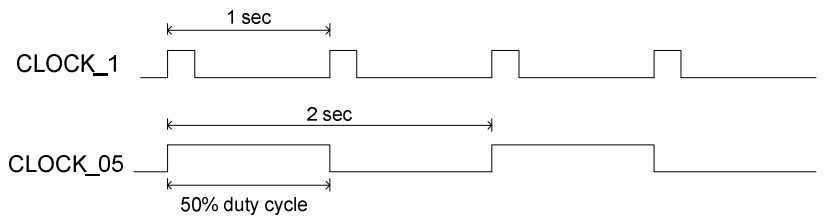
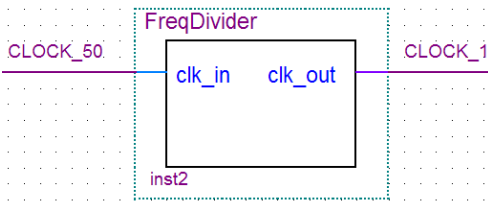


Figure 1. block diagram of a frequency divider module

Figure 2. Timing diagram for *CLOCK_1* and *CLOCK_05* signals

- Bit-synchronous devices use a serial communication protocol to transmit and receive data frames. A data frame typically includes a start-of-frame bit string (*Start*), *N* data bits, and an end-of-frame bit string (*End*). An example of a data frame is shown in Figure 3. In this lab, you are tasked to design a *Start detector* circuit that accepts a serial bit stream (*y*) and detects the *Start* condition. Your design should include an output *z* that is 1 iff the *Start* condition is detected (*z* should remain 1 until the system is reset). Include an active low *resetrn* signal that will reset all flip flops to 0.



Figure 3. An example of a data frame in bit synchronous communication (EC262).

- Draw a state diagram for your design of a *Start detector* circuit.
- Draw a state table for your design.
- Draw a block diagram for your design. Use either DFFs or JKFFs.

LAB

In this lab, you will implement the *Start detector* circuit on a DE2 board.

1. Create a new project and implement the circuit to generate *CLOCK_05* signal (from PRE-LAB #1). Use the necessary components from the Quartus library. Compile your design. **Ask your instructor to verify your circuit.** Create a symbol file for this circuit. This symbol will be used later in LAB #4.

Note: You will need to download and add the following three files to your project (from lab 5).

- *freqDiv.vhd* (no modifications needed).
- *FreqDivider.bdf* (no modifications needed).
- *FreqDivider.bsf* (no modifications needed).

2. Create a new block diagram/schematic file and implement the *Start detector* circuit from PRE-LAB #2.
3. Compile and simulate your *Start detector* circuit. When you run the simulation, use an input clock signal based on *CLOCK_50* (50-MHz). Verify that the designed circuit can detect a *Start* bit string (*red* bit string in the sequence below).

- Simulate with this input sequence for y:

... 0001101100111110*011110*0001111111 ...

Note that each value in the sequence will be available for one clock period.

- When you debug/verify your design in the simulation, it is useful to output the present state that can be observed/monitored in the timing diagram.
4. Modify your circuit to display the output *z* on a 7-segment display (0 or 1). For this implementation, use *CLOCK_05* from LAB #1 as a clock signal to the flip flops. In addition, add an output to *LEDR0* to indicate when an operator/user can change the value of *y* (see requirement f below).

Requirements:

- a. Use *KEY0* for the *resetsn* signal.
- b. Use *SW0* for the *y* signal.
- c. Use *CLOCK_50 (PIN_N2)* for the clock signal to the frequency divider module.
- d. Use *CLOCK_05* for the clock signal to the flip flops.
- e. Use *HEX0* for the output 7-segment display.
 - 0 is displayed if the *Start* condition has not been detected.
 - 1 is displayed if the *Start* condition has been detected. 1 should be displayed until the system is reset.
- f. Use *LEDR0* for an indicator to specify when an operator/user can change the value of *y*.
 - When *CLOCK_05* is high or 1, *LEDR0* is ON, which indicates that an operator can change the value of *y* for the next bit in the sequence.
 - When *CLOCK_05* is low or 0, *LEDR0* is OFF, which indicates that an operator should not modify the value of *y*.

Demonstrate a working circuit to your instructor.