University of Tehran
College of Engineering
School of Electrical & Computer Engineering

Experiment 2
Session 2

Frequency Regulation

Digital Logic Laboratory
ECE 045
Laboratory Manual

Spring 1399

Contents

Introduction

Synchronization is a crucial issue in sequential digital circuit design. Many electrical devices devote multiple internal clocks to synchronize the internal processes. As you got familiar with the clock generation concepts in the previous experiment, it is consisted of a reference clock generator, like a ring oscillator, that is desired to output a stable reference clock by putting an odd number of inverters in a loop chain. However, the thermal variation of the oscillator causes variations in the clock frequency and this ruins the calculations in a digital processor.

The goal of this experiment is to design an adjustable clock generator that can fix the output frequency at the desired value. You will learn how to consider the hardware design to make it a synthesizable one.

By the end of this experiment, you should have learned:

- The concept of adjusting clock frequency
- Hardware design
- Writing test-bench and simulation

The block diagram of the clock adjustment system is shown in figure 1. It is consisted of two parts: The first one is the circuit used in the previous experiment including a ring oscillator, a counter and a T-flip flop for frequency division. The second part is a processing element that receives the output of the flip-flop and sets its frequency to a specific value via a feedback path. In fact, the processor changes the value that the oscillator is generating by dividing it by either a larger or a smaller value.

You are to write a Verilog code for the processing element in FPGA.Then you should model the system behavior in Modelsim by writing an accurate hardware model in simulation and verify your design.

1 Design Description and Synthesis

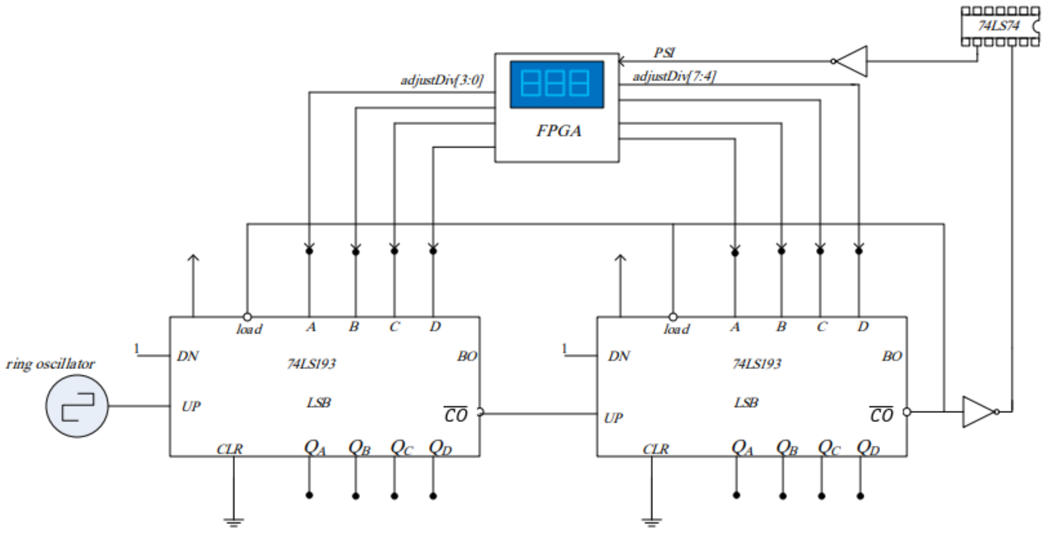
The goal of the design is to receive the output of frequency divider and set the output frequency at a fixed value. To do this, a design like figure 1 is needed. The processor should calculate the input frequency and compare it with a desired reference signal. If the reference and the input's frequencies are not the same, then it should change the division value. So, the load inputs of the counter should be fed by the FPGA and must be increased or decreased whether the frequency is higher or lower than the reference value.

The inputs and outputs of the design are shown in figure 2. `PSI`, the divided clock after passing flip-flop, is the main input of FPGA and frequency regulation will be performed based on it. The adjusted value for division (counter load values) after processing is called `adjustedDiv`. Cyclone IV board works with a 50 MHz clock frequency that is definitely much higher than the input frequency. When the input comes in, FPGA starts to count the input time duration by this clock at the input rising edge and stops counting by the falling edge.

A reference value called `setPeriod` is also another input that represents the desired output frequency in terms of number of clock cycles in one time duration.

To measure the input frequency, a counter starts to count the signal duration when the input signal gets 1. When the input goes to 0, the counter stops counting and the last value of the counter

Figure 1: Clock adjusting system



will be stored in a register. At the rise edge of the input signal, the counter value will be reset. Use an `always` statement like below to determine the duration value. You can use a `case` statement to set the duration value at 4 states described above:

```
always @(posedge clk, posedge rst) begin : decide_when_to_count_and_count
    if (rst)
        // ...
    else begin
        case (/* input signal transition */)
            // zero to one
            // steady at one
            // one to zero
            // steady at zero
        endcase
    end
end
```

Figure 2: Frequency regulator pins

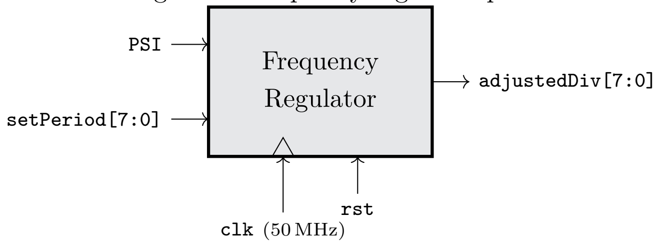
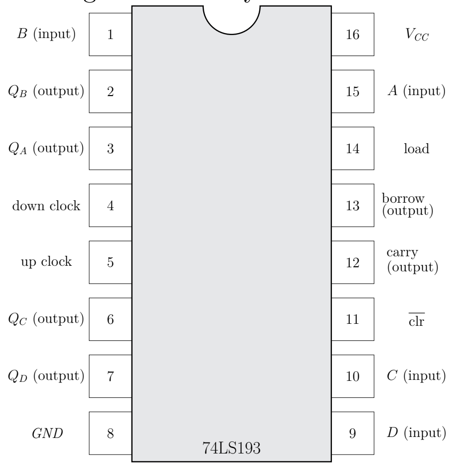


Figure 3: Pin layout of 74193



The comparison must be performed when the input falls to 0. Frequency adjustment will be performed by an increment or decrement signal. When the duration is more than the reference value, the present value of the load inputs should be increased and if its value is less than the reference signal then the load inputs should be decreased. Hence an Increment and decrement signal should be set to 1 after the comparison.

Again, use an `always` statement to change the increment and decrement value at the proper time:

```
always @(/* input and count transition */) begin : comparison
    if (/* one to zero */) begin
        // comparison
        // set the inc and dec flag value
    end
end
```

When the comparison is completed, by the falling edge of the input signal and based on the increment or decrement value, the present value of the load inputs will be changed and registered to the adjusted value. The third `always` block is dedicated to this performance:

```
always @(posedge clk, posedge rst) begin : increment_decrement
    if (// one to zero //) begin
        // increment or decrement
    end
end
```

1. Write a Verilog code based on this instructions.
2. Synthesize this code as a top-level entity.
3. Create a symbol for this frequency regulator module.
4. Change the block diagram of experiment one that includes the counter divider and the T-flip flop. For this purpose, you should use 74LS193 instead of 74LS191 based on the connections of figure 1. The pin layout of 74LS193 is shown in figure 3.

Table 1: Parallel loads of example frequencies

Ring Oscillator Frequency	Desired Frequency	Final Parallel Loads	Initial Parallel Loads	Setperiod
20 MHz	400 kHz	205	127	125

5. Add the frequency regulator symbol to this block diagram.
6. set the necessary inputs and outputs. Note that the Frequency Regulator module needs a a 50 MHz clock signal that is separate from the divider clock.
7. Synthesize the design.
8. Include all the synthesis results, the Quartus II files in your report.

2 Design Simulation in Modelsim

After synthesizing your design, you should verify the hardware.

1. Provide a test-bench for you design and verify your design. You should accurately model the real hardware inside your test-bench. To do this, like experiment one, use the ring oscillator with a frequency smaller than 50 MHz (near 20 MHz).
2. In your testbench, first start with approximately 20 MHz ring oscillator frequency. Then after proper time periods, change this frequency in small range near 20 MHz and verify the regulator performance. to report the results, fill the table 1 like the example. Keep the desired frequency at 400 KHz and change the ring oscillator frequency at different time periods and write the corresponding achieved parallel loads in the table.
3. In your report, include the Modelsim waveforms for these signals: duration, increment, decrement, setperiod, Ring oscillator clock, 50 MHz clock, PSI, and adjusteddiv.

Acknowledgment

This lab manual was prepared and developed by Katayoon Basharkhah, PHD student of Digital Systems at University of Tehran, under the supervision of professor Zain Navabi.

This manual has been revised and edited by Hadi Safari, undergraduate student of Computer Engineering at University of Tehran.