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

Experiment 1

Sessions 1, 2

Clock and Periodic Signal Generation

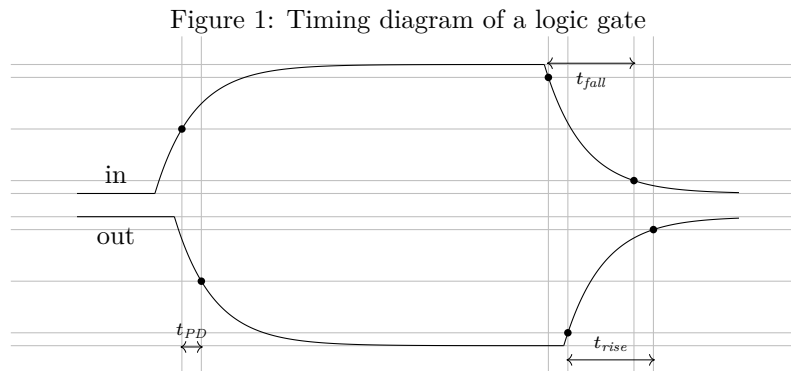Digital Logic Laboratory

ECE 045

Laboratory Manual

Spring 1400

Contents

Figure 1: Timing diagram of a logic gate

Introduction

The goal of this experiment is to introduce the concepts of static characteristics of digital logic gates, delay times, clock frequency generation and digital system using schematic diagram and *Verilog* HDL. This lab is organized in two separate segments. The first segment emphasizes on the analog implementation of different oscillator circuits using LTspice simulator.In the second part you will implement the same circuits in Verilog HDL using Quartus. Finally you will get into FPGA implementation using Altera-Modelsim simulation tool.

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

- Power supply, Function Generator, and Oscilloscope

- 74 Series Basic Logic Gates

- Different oscillator circuits (a LM555 timer IC, Schmitt trigger Oscillator)

- Cyclone IV FPGA Devices

1 Clock Generation using ICs and Analog components

In this part, you will understand different methods of clock generation in digital systems. You will use LTspice simulation tool for implementing the circuits of this section. Also a separate demo on running examples in LTspice will be provided for you.

1.1 Ring Oscillator

In a physical device, no gate can switch instantaneously; in a device fabricated with MOSFETs, for example, the gate capacitance must be charged before current can flow between the source and the drain. Thus, the output of every gate changes a finite amount of time after the input has changed.Therefore, one of the most important parameters in digital logic gates is the propagation delay, that is defined as the time from the 50% point of input to the 50% point of output, as shown in figure 1. There are also two more parameters named t_{fall} and t_{rise} that are measured as the time between 10% and 90% point of the signal and vice versa.

Figure 2: Ring oscillator


Figure 3: LM555 timer pin-out
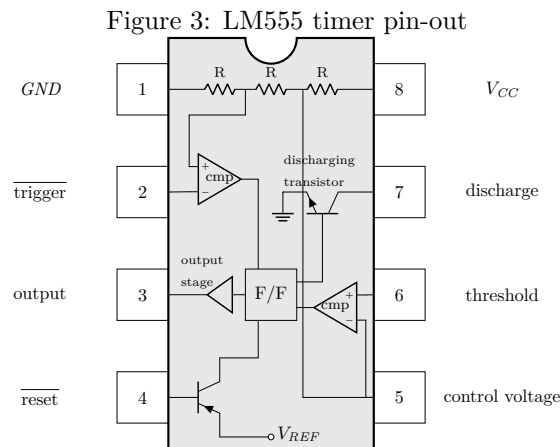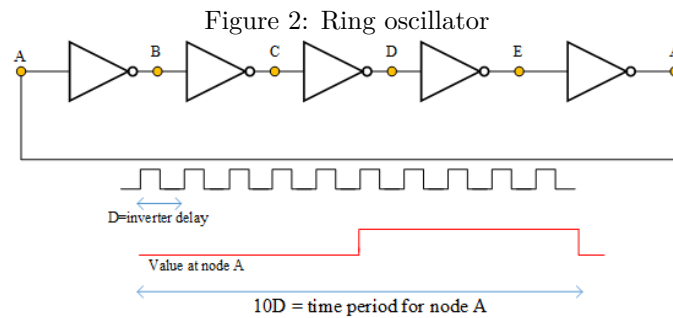
The delay of logic gates is very small and this imposes a large bandwidth. So, measuring this delay directly using a relatively low-cost oscilloscope may be difficult. An alternative method for measuring this parameter is using a *ring oscillator* as shown in figure 2. A ring oscillator is composed of a chain of odd number of inverters, in which output of last inverter is connected to the input of first one. It can be easily seen that adding more inverters to the chain increases the total gate delay. The time at which a value feeds back to the same node is the time period of the ring oscillator which equals $2N * Delay_{inv}$, where N is the odd number and $Delay_{inv}$ is the delay of each inverter gate. The delay of each single inverter can be determined by measuring the total delay. Simulate the circuit shown in figure 2 using inverter gate in 74HCT series (74HCT04).The circuit simulated and output waveform must include in your report.

1. Measure the propagation delay of the chain by measuring the period time of the output.

2. Calculate the delay of a single inverter and report this time.

1.2   LM555 timer

LM555 is among the devices can be used for generating clock signal or time delays. The pin layout of this IC can be seen in figure 3.

Figure 4: LM555 in astable mode
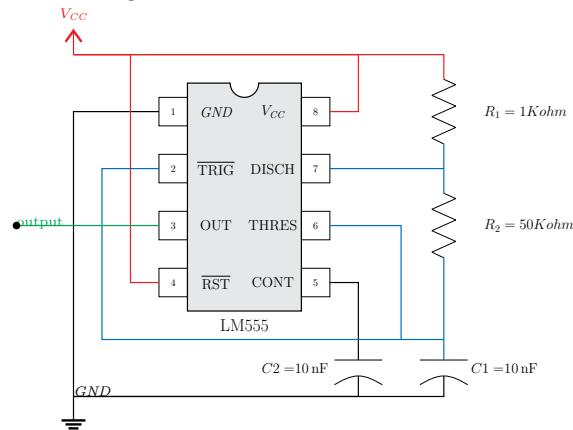
This IC operates in three modes: Monostable, Bistable and Astable. The astable mode that we use in this experiment allows the timer to operate as an oscillator that outputs a continuous rectangular pulse of a required frequency. For astable operation, we need two resistors and one capacitor to design a circuit that operates at the frequency required. The timing during which the output is either high or low is determined by these externally connected resistors and capacitors. Note that the durations of the low and high states may be different. Figure 4 illustrates an LM555 configuration for astable mode operation.

The external capacitor $C1$ charges through $R_1 + R_2$ and discharges through R_2. Thus, the duty cycle and frequency may be precisely set by selecting the right combination of resistances and capacitance. According to figure 3 and figure 4, the charge time (output high) is given by $T_1 = 0.693 * (R_1 + R_2) * C$ and the discharge time (output low) by $T_2 = 0.693 * R_2 * C$. Thus, the total time period of square wave is $T = T_1 + T_2 = 0.693 * (R_1 + 2R_2) * C$. Consequently, the frequency of oscillation is $\frac{1}{T}$. The duty cycle also can be computed by $\frac{R_1 + R_2}{R_1 + 2R_2}$.

These equations describe how we can choose these three values to decide on the frequency and the high and low duration of our signal. With the LM555 timer, the default value of R_1 in this configuration is $1\,\text{k}\Omega$ and this means that we cannot get a perfect 50% duty cycle. (If we make $R_2 \gg R_1$ then we can get close.)

Do the following work. Your report must include the procedure you followed, as well as any observation and results.

1. Implement the LM555 in astable mode using the wiring diagram from figure 4 and observe the output. Report the clock frequency and the duty cycle and include the waveform of the output in your report.

2. Change the value of R_2 resistors to produce different clock frequencies. To do so, R_2 should be $1\,\text{k}\Omega$, $10\,\text{k}\Omega$ and $100\,\text{k}\Omega$. Calculate the frequency and duty cycle using above equations and compare them to the clock signal you see on the output.

Figure 5: Schmitt inverter oscillator circuit
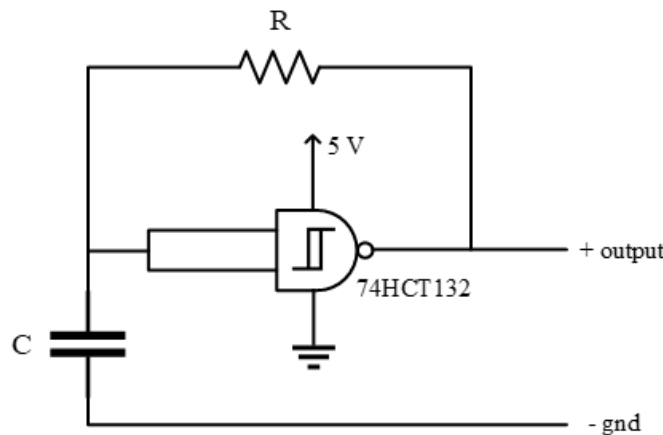
1.3 Schmitt Trigger Oscillator

You can see the realization of the Schmitt inverter oscillator in figure 5. In the Schmitt inverter oscillator, $f = \frac{\alpha}{RC}$, where α is a constant. Study the principle of this circuit and be prepared for it. For schmitt inverter oscillator circuit use 74HCT132.

1. Considering the given equation, try the circuit with different values for the resistor and observe the changes. Use $470\,\Omega$, $1\,\mathrm{k\Omega}$ and $2.2\,\mathrm{k\Omega}$ for the resistor and $10\,\mathrm{nF}$ for the capacitor.

2. Find α parameter.

2 FPGA Design

In this part, you will be familiar with FPGA design. First you will start with simulating different design using 74series IC. In this part you will use Intel Quatus Synthesis tool for mapping your designs on Altera CycloneIV FPGA devices and Modelsim simulation tool for verifying your designs. The FPGA device that you will use in this and all other experiments until the end of the semester is *Cyclone IV* device. Figure 6 shows the layout of the board. In this section you use 74 series ICs. These ICs are placed in Quartus tool inside the Altera MAXPLUS2 libraries. In order to access these libraries and for simplicity, you will design the circuits in a block diagram format. Appendix A shows a guideline for building a schematic design project in Quartus software.

In this part you will use the clock circuits of the first section along with FPGA to simulate a clock frequency divider. There are many parts which should be considered in designing the desired system. Below is a short description of these parts and what you should follow for this experiment.

2.1 Ring Oscillator

You have seen the functionality of a ring oscillator by simulating the circuit in LTspice. Now you will redo the simulation using Verilog HDL code.

Figure 6: Altera cyclone IV board
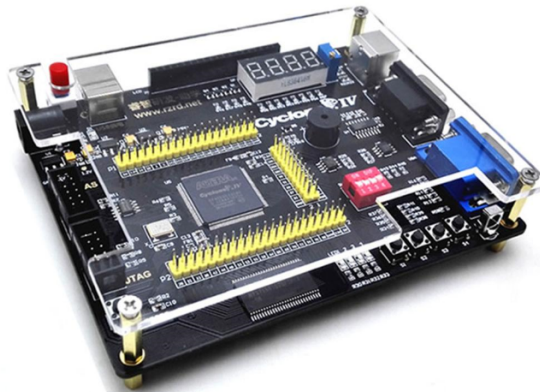
1. Write a Verilog description of the ring oscillator of section 1.1.For this purpose you need to use an odd number of inverters. Your design must be parameterized so that it receives the delay value of the inverters and the number of inverters as parameters and generates the corresponding oscillator as the output.

2. To verify your design, you need a simple testbench. Make an instance of the ring oscillator in your testbench and test the design by providing the parameters. For real modeling of the ring oscillator, in your testbench use the delay values that you calculated in section one.

3. Measure the ring oscillator frequency and include the waveforms of the inputs and outputs in your report.

2.2 Synchronous Counter as a Frequency Divider

Different clock signals can be produced by the aforementioned methods, but not all of them are suitable for all applications. Consider a 1 Hz clock signal which can be easily produced by a LM555 timer. The timing error of this signal can be 1-2%, which is too much for a low frequency like this, while this error range is acceptable for higher frequencies. So, a higher frequency can be chosen and then a frequency divider can reduce the frequency to the desired one.
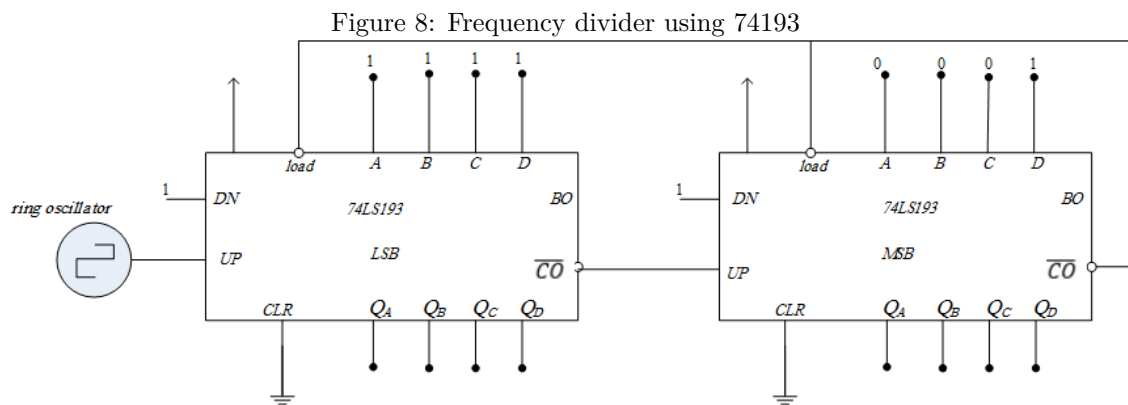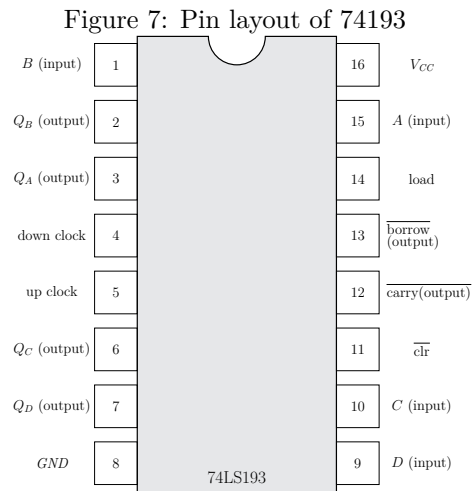
Counters can be used as a frequency divider. 74LS193 is a synchronous 4-bit up/down counter. As figure 7 shows, it has 4 inputs D-A(most to least), 4 outputs Q_D-Q_A (most to least)and a parallel load, which allows for presetting an initial value for the counter. With this pin layout two counters can be cascaded when the modulus is more than 4 bits.
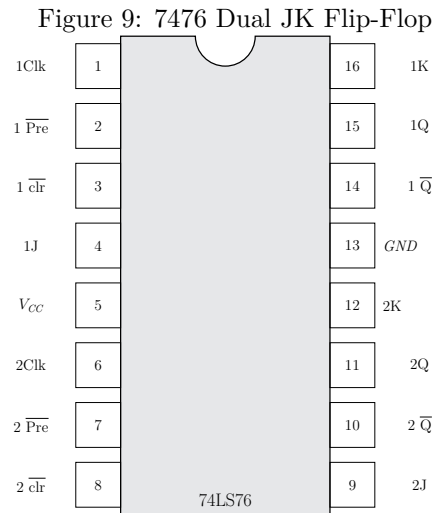
When up counting is desired the initial value is obtained by:

$$Initial\ value = Maximum\ value - Modulus$$

Construct a divide by 113 synchronous up-counter as shown in figure 8. You should:

1. Use the MAXPLUS2 74 series (74193) IC in your block diagram and connect them based on figure 8 and layout offigure 7.

Figure 7: Pin layout of 74193

Figure 8: Frequency divider using 74193

Figure 9: 7476 Dual JK Flip-Flop

1Clk	1		16	1K
$1\overline{\text{Pre}}$	2		15	1Q
$1\overline{\text{clr}}$	3		14	$1\overline{\text{Q}}$
1J	4		13	*GND*
V_{CC}	5		12	2K
2Clk	6		11	2Q
$2\overline{\text{Pre}}$	7		10	$2\overline{\text{Q}}$
$2\overline{\text{clr}}$	8	74LS76	9	2J

2. A Presetting mechanism is necessary for initial loading of the counters of figure 8. The mechanism as you have learned in the logic design course can include an AND gate for anding a preset input signal with the load inputs of the counter. Perform this presetting by using 7408 AND gate of MAXPLUS2 74 series.

3. Write a testbench for the design and set the required input values like the initial load value, enable and up/count signal. for counting up the DNU signal must be set to zero.

4. Use the ring oscillator of the previous part as the clock input of the LSB counter. Since the ring oscillator model is not a synthesizable design, connect the ring oscillator inside the testebench and not in the Quartus block diagram.

5. After simulating the design in Modelsim software, record the results of carry out of the MSB counter and measure its frequency, compare the results with the frequency of the input clock.

2.3   T Flip-Flop

You should use a T Flip-Flop after counter to produce a 50 percent duty cycle signal. Use 7476 IC from the MAXPLUS2 library for this purpose. 7476 is a dual JK Flip-Flop that you must convert it to a T Flip-Flop. Pin layout of this IC shown in figure 9.

3   Baud Rate Generator for UART Serial Communication

UART stands for Universal Asynchronous Receiver/Transmitter. It's a physical circuit in a microcontroller, or a stand-alone IC. A UART's main purpose is to transmit and receive serial data.In UART communication, two UARTs communicate directly with each other. The transmitting UART converts parallel data from a controlling device like a CPU into serial form, transmits it in serial to the receiving UART, which then converts the serial data back into parallel data for the receiving

Figure 10: UART packet data

| 1 start bit	Data Frame	0 to 1 Parity bit	1 to 2 stop bits

Figure 11: Baud Rate Generator
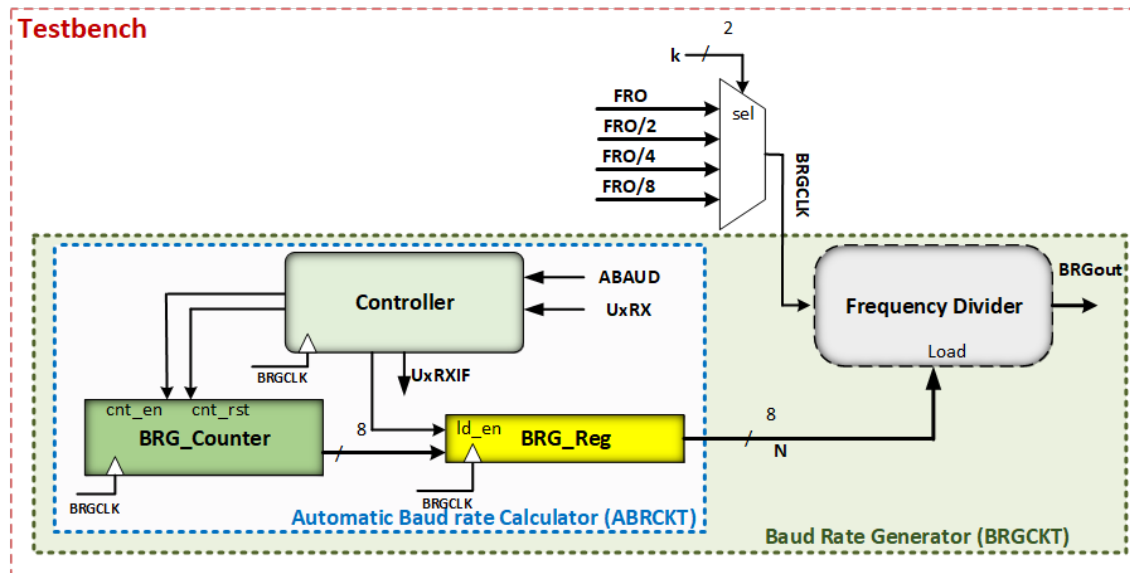
device. UARTs transmit data asynchronously, which means there is no clock signal to synchronize the output of bits from the transmitting UART to the sampling of bits by the receiving UART. Instead of a clock signal, the transmitting UART adds start and stop bits to the data packet being transferred. UART transmitted data is organized into packets. Each packet contains 1 start bit, 5 to 9 data bits (depending on the UART), an optional parity bit, and 1 or 2 stop bits as shown in figure 10.

When the receiving UART detects a start bit, it starts to read the incoming bits at a specific frequency known as the baud rate. Baud rate is a measure of the speed of data transfer, expressed in bits per second (bps). Both UARTs must operate at about the same baud rate. For this purpose a baud rate generator module will be used for generating the desired frequency on the receiver side. The transmission/reception clock is generated by dividing the main clock. The baud rate generated from the main clock is obtained by the following equation:

$$fBaud = \frac{Fclk}{N \times 2^k \times 2}$$

Where k is a scaling factor to generate powered two scales of the main clock before generating the baud rate and can be set to a fixed value between zero to 3 by user. N is the baud rate value which will be stored in a register for baud rate generation. As can be seen this equation can be easily implemented using a frequency divider. This is shown in figure 11. In this part you are to design a baud rate generator using the frequency divider of previous section

Figure 12: Automatic Baud Rate calculation
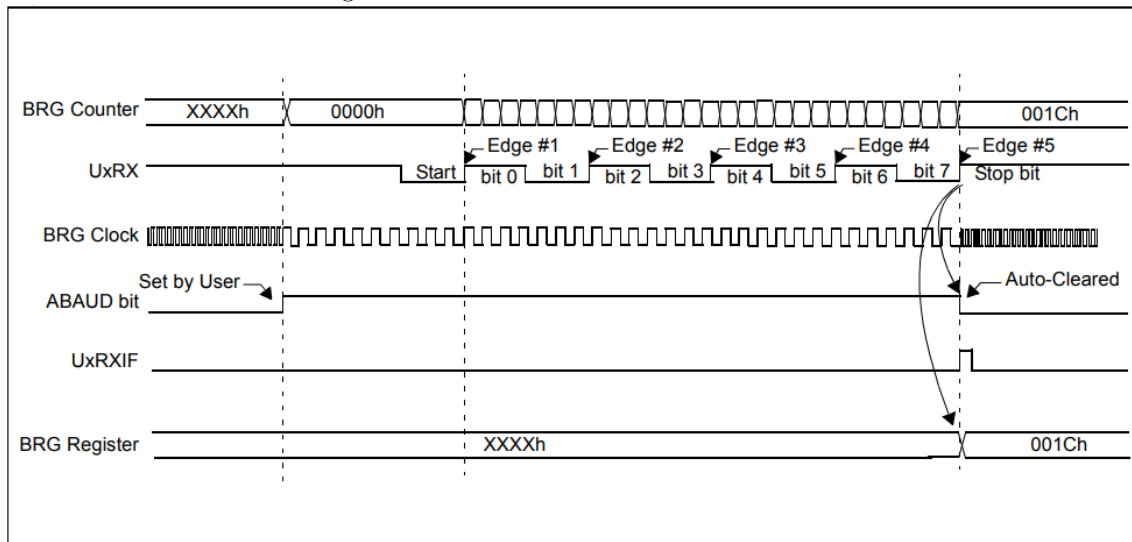
3.1 Automatic Baud Rate Calculator

Although the value of BRG register,N, in figure 11 can be set manually by user, many microcontroller vendors provide their system with an automatic baud rate calculation unit that calculates the value of Baud rate. The functionality of this unit will be explained here based on the timing of figure 12. To allow the system to determine the baud rates of the received characters, the ABAUD bit is enabled. The UART begins an automatic baud rate measurement sequence whenever a Start bit is received, and when the Auto-Baud Rate Detect is enabled (ABAUD = 1). When the ABAUD bit is set, the BRG counter value clears and looks for a Start bit – which, in this case, is defined as a high-to-low transition, followed by a low-to-high transition. Following the Start bit, the auto-baud expects to receive an ASCII 'U' (55h) to calculate the proper bit rate.At the end of the Start bit (rising edge), the BRG counter begins counting up. On the 5th UxRX pin rising edge, an accumulated BRG counter value totaling the proper BRG period is transferred to the BRG register.The UxRXIF is an interrupt flag that is set on the 5th UxRX rising edge and used for microcontroller operations.

1. The main part of the Automatic Baud Rate Calculation unit, ABRCKT, is its controller that generates the required sequence and timing of figure 12. Show a state diagram of the controller and write its Verilog description.Use the Huffman style coding for the controller.

2. Write the Verilog description of the datapath of ABRCKT unit.

3. Integrate the datapath and controller unit in a top-level module named ABRCKT.Set this module as the top-level entity and synthesize it in Quartus.

4. Generate a schematic symbol for this module.

5. Connect the ABRCKT module to the frequency divider module of section 2 in a block diagram file named BRGCKT according to figure 11.

6. Set back the BRGCKT block diagram as the top-level entity and synthesize it.

7. Write a testbench for the total design and set the required inputs and outputs. Make an instance of BRGCKT inside the testbench. Generate BRGCLK in your testbench. This can be done both in a functional or structural model. In this model you need to use the ring oscillator that you generated in section 2. Based on the scaling factor you can create four different scaled version of the ring oscillator clock. Write and test your design in *ModelSim* and check if it is working correctly.

8. For testing the design consider a fixed value of k and generate a wave for UxRX input as shown in figure 12. Verify the correctness of your design by calculating BRGout frequency both from simulation and theory.

Appendices

Appendix A   Using Quartus II

A.1   Create the Project

1. Click on *File* ▷ *New Project Wizard*

2. Create an appropriate directory for your project and complete the form

3. Select the FPGA device as *Cyclone EP2C20F484C7* and then click *Finish*

4. From *File* ▷ *New* select the *Veriog HDL File*

5. Write your Verilog code in this window

A.2   Compilation

- Select *Processing* ▷ *Start Compilation*

- Click on quick shortcut ▶

There may be lots of warnings and some errors after compilation. The warnings are not so important while the errors should be completely removed.

A.3   Pin Assignment

After you successfully compile your design, you should set your design with physical pins of the Cyclone II FPGA on DE1 board. You can find the position and the index of each pin from the DE1 user manual.
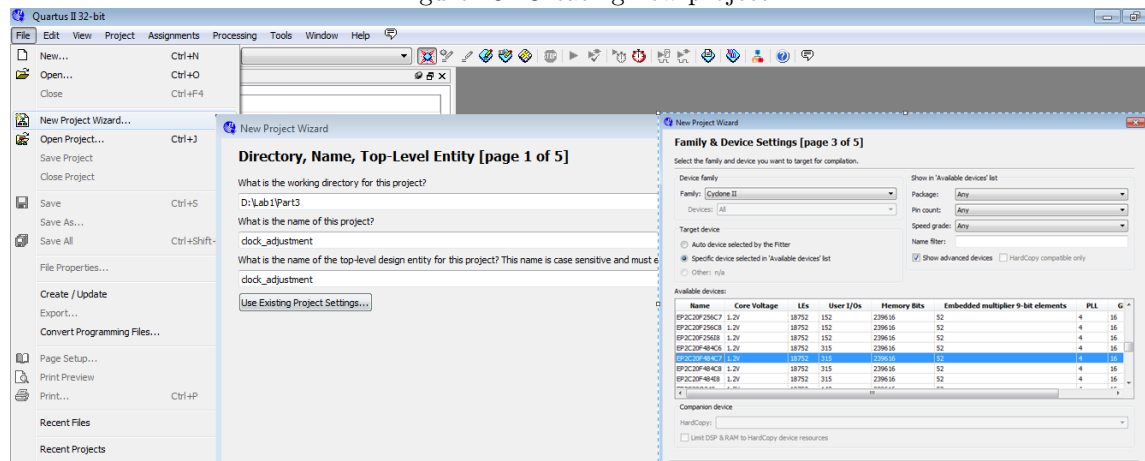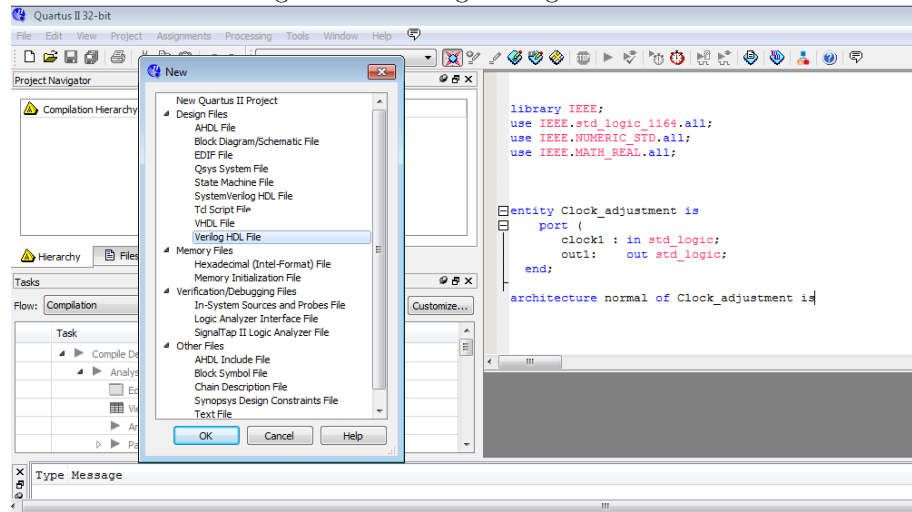
Figure 13: Creating new project

Figure 14: Writing Verilog HDL code

From *Assignments ▷ Pin Planner*, Pin Planner window will appear and you can select the corresponding locations from the list. When you are done with this, recompile your project.

A.4   Program the Design

1. Click on the ✋ icon on the toolbar to open *Programmer*

2. Choose *USB Blaster* from the *Hardware setup* in the new window

3. Click on the start button

Your design will be programmed on the board when it is finished.

A.5   Examine the Timing and Resources

Now you can examine the resource usage of your design from *Processing ▷ Compilation Report* or by clicking on quick shortcut ✋, and the timing from the *Tools ▷ TimeQuest Timing Analyzer*.

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.

Figure 15: Pin Planner window