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

Experiment 4
Sessions 10,11,12

Integrated System

Digital Logic Laboratory
ECE 045
Laboratory Manual

Spring 1400

Contents

Contents **1**

Introduction **2**

1 Frequency Multiplier **3**

2 Exponential Accelerator **4**

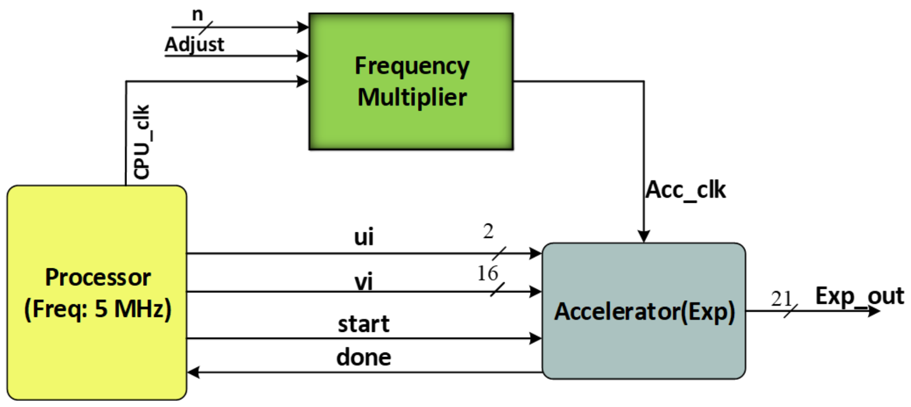
2.1 Exponential Engine 4

2.2 Exponential Accelerator Wrapper 6

3 Integrated Circuit **8**

Acknowledgment **8**

Figure 1: Block diagram of a typical integrated circuit



Introduction

System on Chip is an integrated circuit that integrates multiple components including digital, analog, hardware and software programs all in a single chip. The main core of an SoC is a processor that handles different computational tasks within the system. In addition to the processor, the system includes a memory, Input/Output ports and accelerators.

As you have learned from the Digital Logic Design course, accelerators are dedicated computation units that usually execute one specific task. This single task, needs a smaller and less complicated datapath which leads to a high frequency of operation for the accelerators. This is in contrary to CPUs in which millions of operations must be executed within a fix time interval. This impose a low frequency of operation for CPUs. To increase the speed of an SOC, hardware accelerators are usually embedded in the system. The processor will dispute some of its tasks to hardware accelerator.

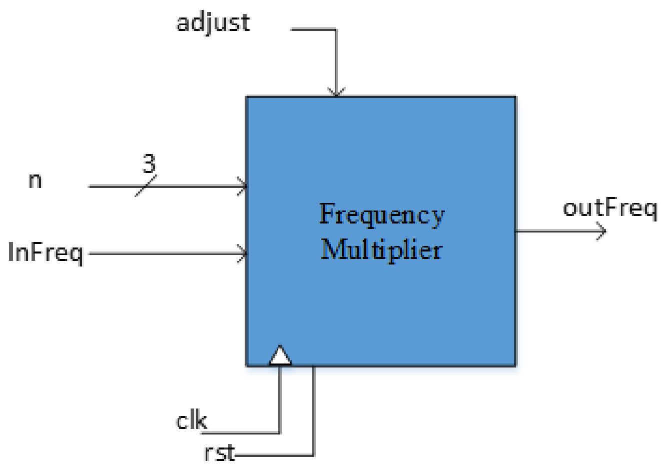
Since hardware accelerators have a higher frequency than the processor, there is a need for a frequency multiplier to multiply the processor frequency and feed it to the accelerators. In this way, while the processor is executing some sequential operations, the accelerator can execute multiple of operations with a higher frequency. This frequency multiplier is usually implemented with a Phased Lock Loop in real integrated systems. In this experiment you are going to design a frequency multiplier and use this in combination with an exponential accelerator.

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

- The concept of an SOC
- The concept of handshaking in an SoC
- The principle of an accelerator
- Processor and Accelerator timings

Figure 1 shows the block diagram of a typical integrated system including a processor and an accelerator.There is a handshaking between these two components via signals ”start” and ”done”. The processor works with a frequency of 5 MHz. The accelerator which is an exponential circuit

Figure 2: Block diagram of the frequency multiplier



can work with much higher frequency clock signal. This high frequency clock will be generated by a frequency multiplier module. Multiplication factor depends on the target accelerator that the clock is being generated for. There is a maximum frequency for each accelerator that is dominated by its worse case delay time. Considering this maximum frequency, the factor of frequency multiplication can be determined.

Accordingly, Below is the topics that are explained in the following of this experiment in details:

- Exponential Accelerator
- Frequency Multiplier Module
- Integrated System

1 Frequency Multiplier

In this part, you are to design a frequency multiplier.Block diagram of this module is shown in Figure 2. A frequency multiplier takes an input signal frequency (InFreq with frequency f) and multiplies it by a value determined by multiplication factor (MultFactor). The output is a signal (OutFreq) with a frequency equal to InFreq*MultFactor. Multiplication factor is a power of two and could be shown as 2^n, where n is between 1 and 5.Therefore the circuit takes $log_2 MultFactor$ as the input value. For example if MultFactor is 8, the corresponding input value would be 3.

$$f \times \ 2^n = 150 \div k$$

InFreq frequency range is between 1MHz and 50MHz. A reference clock input is used for reference working clock of this circuit and is provided via an input named clock RefClk, with a high frequency of 150 MHz. The output of the circuit that carries the faster signal is outFreq. The frequency of this signal is $f \times \ 2^n$. The circuit has a valid output that is asserted when frequency multiplication is appearing on outFreq. When a positive pulse appears on ”adjust”, the circuit is informed of a new multiplication factor, and begins preparation for the multiplication process with this new factor. While this preparation is taking place, ”valid” becomes 0 and remains 0 until preparation is complete at which time the generated outFreq signal represents the new multiplication factor. For this design, the following equation is true:

For this you need first to calculate ”k” value from the equation above. Remember that you need hardware components like counters and shifters to calculate this factor. After that, you can use factor ”k” for dividing the high frequency (150MHz) reference clock and implementing the right hand side of equation above.

1. Show a state diagram of the controller.

2. Write the complete Verilog description of this design.

3. In your top-level module, dedicate a part for the controller and one for the datapath.

4. Use Huffman coding style for the controller.

5. Use an asynchronous reset and a rising edge clock.

6. Write a testbench for your top-level design.

7. In your testbench provide different test sequence including at least three different values for input ”n”. show and calculate the output frequency from the simulation results.

8. Run the testbench in Modelsim simulation software and include the waveforms and results in your report.

2 Exponential Accelerator

2.1 Exponential Engine

The accelerator that you are going to use is an exponential circuit . You are familiar with this accelerator design.As Figure 3 shows, this module receives a 16-bit input ”x” and generates an 16-bit output ”Fractionalpart” and 2-bit ”Integerpart”. Remember that the x value fits within zero and one. The accelerator starts working with a complete pulse on signal ”start” and when the computation is completed signal ”done” will be sent to the processor to acknowledge it. For the purpose of clock generation for this module, first you need to explore the design accuracy. Furthermore you as a designer need to be aware of the maximum frequency of this accelerator. The Verilog description code for this module is provided to you.

1. First examine the code and its accuracy by running Modelsim simulation. For this purpose, write a testbench for this design with at least three different values for input ”x”. Show the results by taking picture of the simulation results.

2. Synthesize this design in Quartus II Software. Show the synthesis results in your report.

3. After synthesizing design, you can find out the maximum frequency of this accelerator by referring to the Timing Analyzer reports in Quartus synthesis tool. You can follow the steps shown in the Figure 4

Figure 3: Block diagram of exponential accelerator

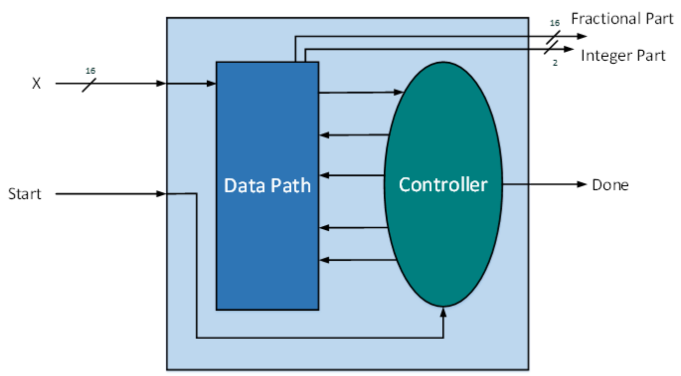
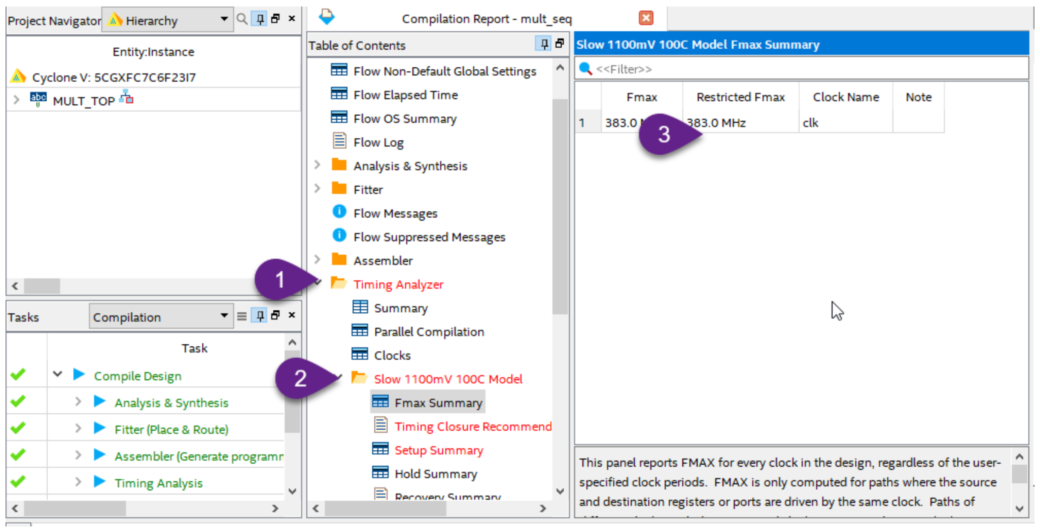


Figure 4: Steps for observing maximum frequency



	Fmax	Restricted Fmax	Clock Name	Note
1	383.0	383.0 MHz	clk	

This panel reports FMAX for every clock in the design, regardless of the user-specified clock periods. FMAX is only computed for paths where the source and destination registers or ports are driven by the same clock. Paths of

2.2 Exponential Accelerator Wrapper

Although the accelerator is working with a higher frequency than the processor, for the handshaking signals of ”start” and ”done” the accelerator have to wait for the processor to send and receive these signals with its low frequency. This imposes some timing overhead to the accelerator and hence performance reduction. In order to use this free time, the accelerator can calculate multiple exponential values. One of the applications that makes use of such multi-value exponential calculation is an activation function in Deep Neural Networks (DNN).

$$f(x_i) = e^{x_i}, \{i = 1, 2, ..., N\}$$

Multiple of these exponential values can be calculated with one accelerator instead of using one accelerator unit for each xi. In order to reduce the required hardware resources for softmax exponential function, mathematical transformation would be useful. Each input value is split into an integer number zi and a fractional number vi as below:

$$x_i = z_i + v_i$$

so

$$e^{x_i} = e^{z_i}.e^{v_i}$$

The second segment of this equation can be easily implemented with the exponential engine. In order to reduce the complexity of the hardware implementation of this equation, we use the base number 2 to take place of the base number e and the exponential function will be changed to:

$$e^{x_i} = 2^{u_i}.e^{v_i}$$

or

$$e^{x_i} = e^{v_i} << u_i$$

where

$$u_i = \left\lfloor z_i.log_2 e \right\rfloor$$

As can be seen, this equation can be implemented with a shifter that shifts e^{v_i} for u_i times.

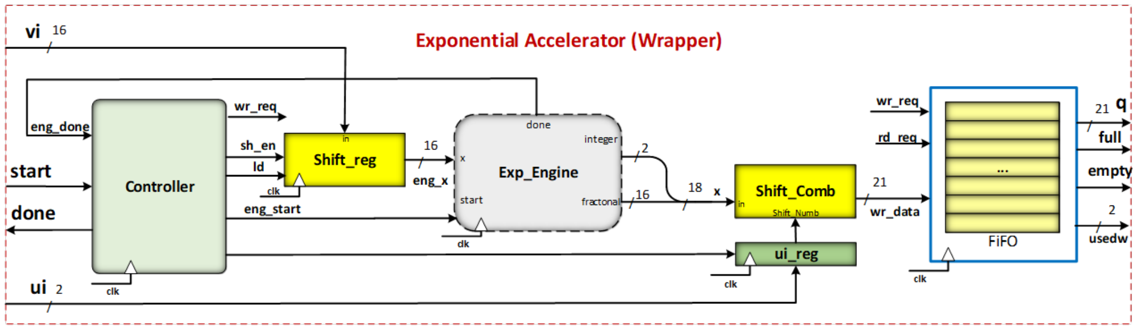
We are going to apply this transformation to the exponential in a wrapper around the exponential engine. Some other tasks are also included in the wrapper that are explained below:

- Referring to Figure 5, the wrapper receives single input in the form of a fractional value, v_i, and an integer value u_i and a start signal from processor.
- Considering the input values are within u_i and u_i+1 we can calculate n number of exponential in this range as follows:

$$e^{x_i} = \begin{cases} u_i + v_i & \\ u_i + 2 * v_i & \text{if } v_i < 1/2^{(n-1)} \\ ... & \text{if } u_i < x_i < u_i + 1 \\ u_i + 2^{(n-1)} * v_i & \end{cases}$$

  In this experiment n=4. Based on this assumptions, 4 exponential values can be calculated. Four different values can be generated with the shift register unit by first registering the value of v_i and then shifting its value one bit to the left for each exponential calculation.

Figure 5: Wrapper for exponential accelerator



- The controller is responsible for generating the load and shift enable signals for shift register, the start signal for exponential engine and the load signal for the ui-register. In fact, the exponential engine should start each calculation when the previous one is completely done. For this purpose *engdone* is fed to the controller and when done is asserted the controller generates a complete pulse on *engstart*. At the same time, the correct value of x should appear on the corresponding input of exponential engine. When all calculations are finished the controller sends a done signal on the wrapper output and also issues the wrreq signal for writ ting data to the FIFO.
- For shifting the output of exponential engine, a combinational shifter is required. The input of this shifter is the u_i value that is provided outside the wrapper considering that $u_i = \lfloor z_i.log_2 e \rfloor$. To store the value of u_i, a register called *uireg* is used.
- When an exponential value is calculated then it should be stored in a FIFO so when the CPU finishes it's work it can retrieve all the results. The FIFO which stands for First Input First Output is an storage element like a memory array that automatically keeps track of the order in which data enters into the module and reads the data out in the same order. As shown in the Figure 5, the FIFO has a *writereq* input and a write data for writing into the buffer and a *readreq* for reading the outputs. In this experiment you will use the FIFO IP provided in Quartus Megawizard. All you need is to connect the proper signals. since we are limited to four calculation in this experiment the FIFO size would be four as well.

1. Show the state diagram of the controller and write the Verilog description of this module in a Huffman style.
2. Write a Verilog description for the wrapper shown in Figure 5.
3. Write a testbench and make instance of this wrapper in your tesbench.
4. At first provide u_i and v_i value and also generate a complete pulse on the signal "start" for the accelerator wrapper.
5. Test your design for at least 2 values of u_i and v_i. Include expected and achieved results in your report and make a comparison.

6. After synthesizing design, find out the maximum frequency of this accelerator wrapper by referring to the Timing Analyzer reports in Quartus synthesis tool. You can follow the steps shown in the Figure 4

3 Integrated Circuit

In this part, you are to connect the accelerator and the frequency multiplier in an integrated circuit. For this purpose connect this two module based on the Figure 1.

1. Connect the circuits of parts 1 and 2 in a top level design. The output of Frequency Multiplier would be the working clock of the accelerator.
2. You have find out the maximum frequency of the exponential accelerator in part 2. Based on this frequency and the frequency of the processor (5MHz), find the proper value range for ”MultFactor” and input signal ”n” in the frequency multiplier module. For example if the maximum frequency is 300 MHz then the ”MultFactor” can take values up to 60.
3. Based on the achieved n value, decide on the maximum exponential calculations that can be supported by your system. Change the FIFO Size and also the controller to implement that number of calculations.
4. Write a testbench and model the behavior of the processor in your tesbench. For this, you need to generate a CPU clock of 5 MHz. At first generate a complete pulse on signal ”Adjust” and then a complete pulse on the signal ”start” for the accelerator. Then provide the value for inputs u_i and v_i. All these data and control signal values should be generated based on the CPU Clock.In this way, the frequency multiplier with the high frequency of 150 MHz generates the ”Acc-clk” and accelerator starts working when receives start.
5. Inside your tesbench make a instantiation of the integrated top level design. Note that you should make an instance of the synthesized top-level design and include .sdo files in Modelsim simulation.
6. Test your design for different values of n within and outside the proper range of accelerator frequency. For example if the n is up to 60, provide values less than and also greater than 60 and verify the output result for all the cases.

Acknowledgment

This manual has been revised and edited by Maryam Rajabalipanah, PHD student of Digital Systems at University of Tehran.