UNIVERSITY OF TEHRAN
Electrical and Computer Engineering Department
ECE 367
Digital Logic Circuits – Spring 1398-99
Computer Assignment 6
RTL Processing Element Design and Implementation
Week 21

Taylor series expansion is one of well-known methods to compute mathematical functions such as *sin(x)*, *cos(x)*, *exp*, *tanh*, etc. In Machine Learning (ML) applications *tanh(x)* plays an important role as an activation function. In hardware implementation of various types of neural networks and creating accelerators for such applications, hardware implementation of this function becomes important.

Among the various implementations of *tanh()*, a relatively expensive with high accuracy is implementing its Taylor series expansion.

In this problem you are to design a sequential circuit that computes an approximation of *tanh()* using the first 8 terms of its Taylor expansion.

$$\tanh x = \sum_{n=1}^{\infty} \frac{B_{2n} 4^n \left(4^n - 1\right)}{(2n)!} x^{2n-1} \qquad = x - \frac{x^3}{3} + \frac{2x^5}{15} - \frac{17x^7}{315} + \cdots \qquad \text{for } |x| < \frac{\pi}{2}$$

Shown below is the block diagram of the module approximating *tanh()*. The module accepts a 16-bit fixed point value on *x* (consider $0 \le x < \pi/2$) after *start* is asserted. After that, computation is started. After the completion of the computation, the result becomes available on output *y*, and *ready* is issued.

Assume $0 \le x < 1$, and all numbers are represented in 16-bit fixed point format. In addition, a 16-bit fixed point adder and a 16-bit fixed point array multiplier are available for you to use as datapath components. Moreover, 16-bit values for the first eight coefficients for the terms of the Taylor expansion series have been computed, and are stored in a combinational lookup table (ROM). The table has three address lines and a 16-bit data output.

A) Generate a ROM in Quartus II for the required memory of factorial calculations. For this part, make sure the way you describe your ROM synthesizes to FPGA memory blocks.
B) Design the datapath of module *tanh(x)*, and implement it in Quartus II. You have the option of using Quartus parts from its library (e.g., lpm), or writing the components in SystemVerilog.
C) Write a Verilog description for the controller of module *tanh(x)*.

D) Complete the datapath and controller of your circuit and develop a testbench for it.
E) Synthesize the description of *tanh(x)*, and create a post-synthesis output.
F) Use the testbench of Part D to simulate pre- and post-synthesis descriptions of your hardware. Show the results and compare timings.