Reconfigurable Computing (CSG533)

Project Report

Hardware Implementation of finding n^{th} Root of a number using Verilog

Ganesh Prasad B K (2018H1230151G)
Siddanth Jain (2018H1230177G)
Group: B3GP2

M.E Microelectronics
Department of Electrical & Electronics
BITS Pilani – K K Birla Goa Campus
Zuarinagar, Goa 403726

Title:
Hardware Implementation of finding n^{th} Root of a number using Verilog.

Objective:
To implement Newton-Raphson method of finding n^{th} root of a number on ZedBoard using Verilog HDL.

Software Used:
Xilinx Vivado 2016.2, ModelSim PE Student Edition 10.4a

Hardware Used:
ZedBoard Zynq-7000 AP SoC XC7Z020-CLG484

Block Diagram:

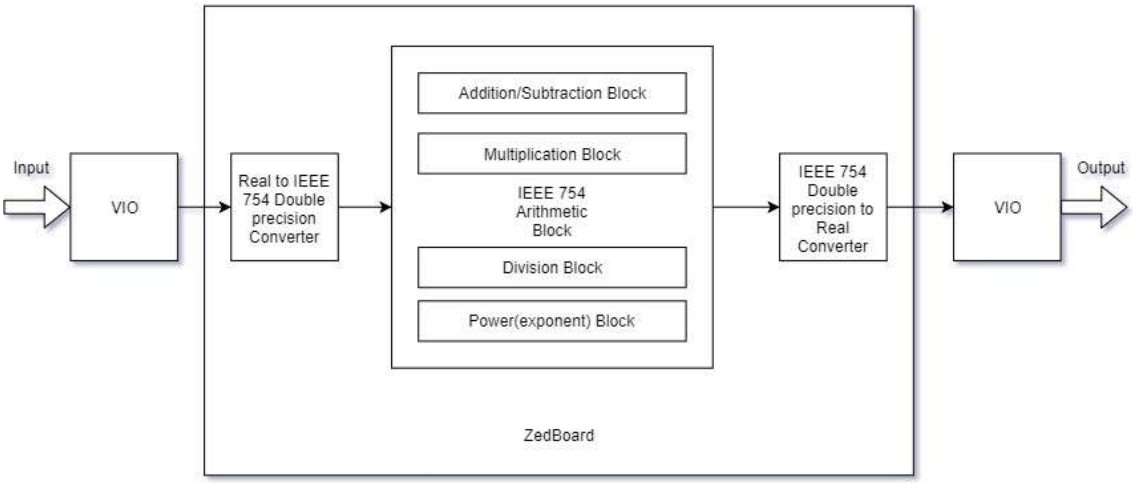

Figure 1. Block diagram for finding n^{th} root of a number using Newton Raphson method

Description:

In numerical analysis, Newton's method (also known as the Newton–Raphson method), named after Isaac Newton and Joseph Raphson, is a method for finding successively better approximations to the roots (or zeroes) of a real-valued function. It is one example of a root-finding algorithm.

x: f(x)=0.

The method starts with a function f(x) defined over the real numbers x, the function's derivative f ′(x), and an initial guess x_0 for a root of the function f(x). If the function satisfies the assumptions made in the derivation of the formula and the initial guess is close, then a better approximation x_1 is

$x_1 = x_0 - \frac{f(x)}{f`(x)}$

Geometrically, $(x_1, 0)$ is the intersection of the x-axis and the tangent of the graph of f(x) at $(x_0, f(x_0))$.

The process is repeated as

$x_{n+1} = x_n - \frac{f(x_n)}{f`(x_n)}$

until a sufficiently accurate value is reached.

<u>Deriving Newton Raphson Method iterative formula for finding to nth of a number 'a':</u>

$x = \sqrt[n]{a}$

$\rightarrow x^n = a$

$\rightarrow x^n - a = 0$

Let $f(x) = x^n - a$

$\rightarrow f'(x) = nx^{n-1}$

Substituting above equation in Newton's iterative formula $x_{k+1} = x_k - \frac{f(x_k)}{f`(x_k)}$

we get $x_{k+1} = x_k - \frac{x^n - a}{nx_k^{n-1}}$

on simplification we get

$$x_{k+1} = \frac{1}{n}\left[(n-1)x_k + \frac{a}{x_k^{n-1}}\right] \qquad \text{Eqn. (1)}$$

The above equation is implemented on ZedBoard using various modules to get the nth root of a number.

Inputs are given through VIO and outputs are observed on the VIO itself.

<u>IEEE 754 Floating Point Representation:</u>

The IEEE 754 standard for Floating Point representation is a technical standard for floating point computation The standard addressed many problems found in the diverse floating point implementations that made them difficult to use reliably and portably. IEEE 754 Floating point represntation is for non integral numbers including very small and very large numbers.

Universally adopted two representations are

Single Precision (32 bits)

Double precision (64 bits)

IEEE 754 Floating point format:

	Single: 8 bits	single: 23 bits
1 bit	double: 11bits	double: 52 bits
Sign	Exponent	Fraction

$$X = (-1)^{Sign} + (1 + Fraction) \times 2^{(Exponent - Bias)}$$

Sign bit i.e., 0→non negative, 1→negative

Exponent: Excess Representation i.e., Actual Exponent+Bias (Single precision bias is 127 and double precision is 1023)

Flow Chart:

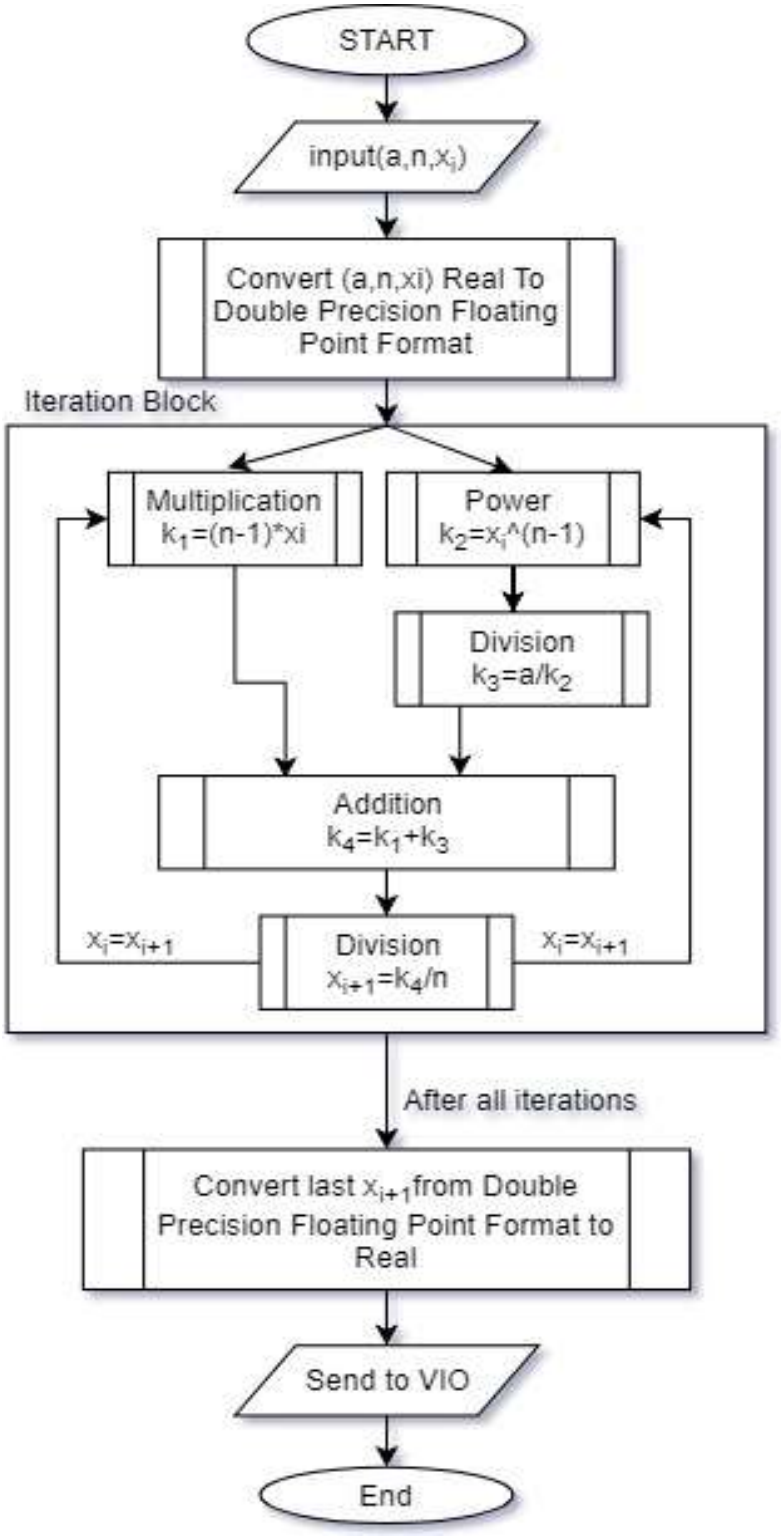

Figure 2. Flow Chart for finding n^{th} root of a number using Newton Raphsons Method

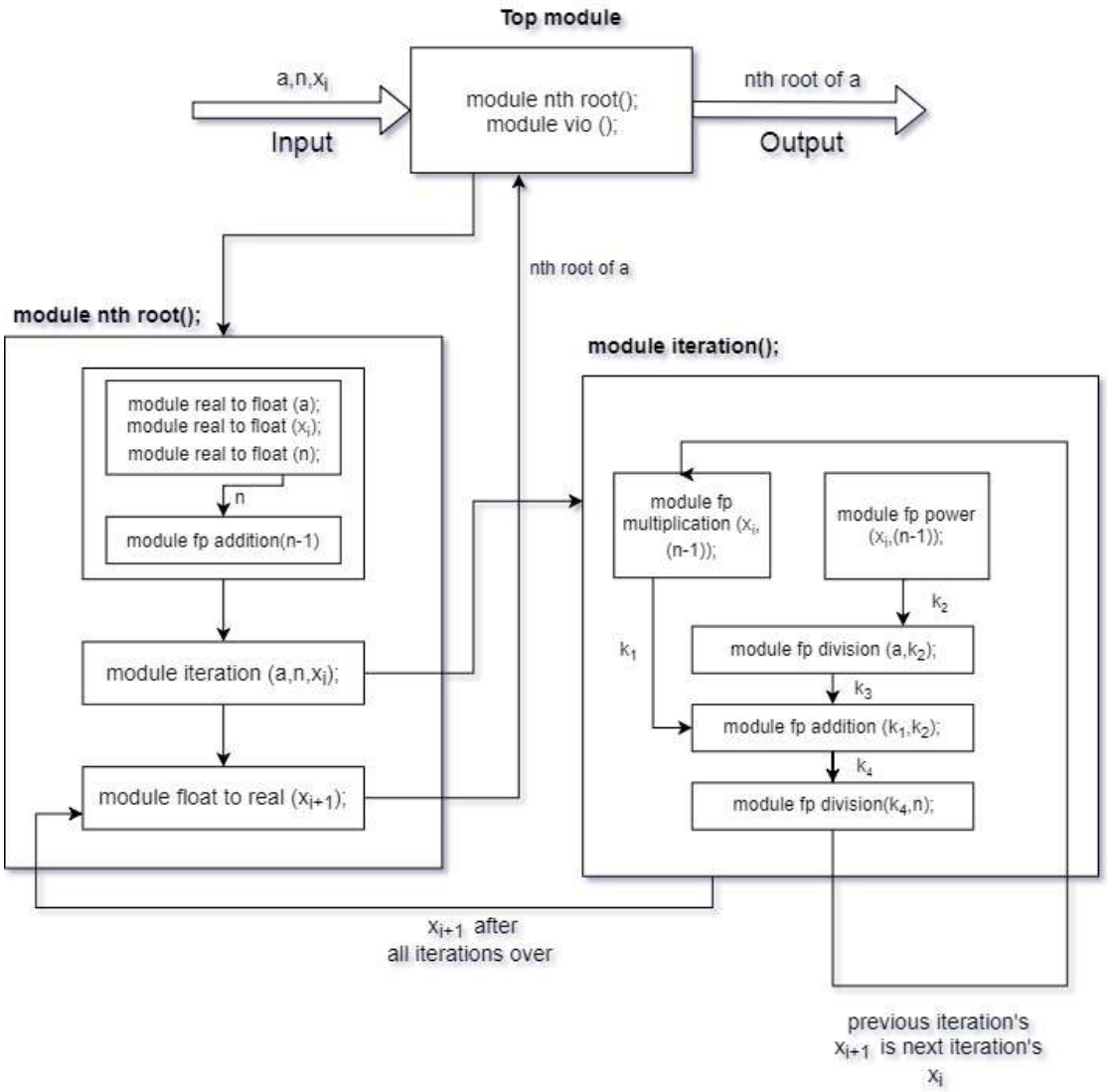

Figure 3. Verilog Implementation flow

From Figure 2 andFigure 3

1. a, n, x_i are inputs where, a is the number whose n^{th} root has to be found and x_i is the initial guess.
2. The inputs entered in the real format i.e., 32.32 are converted to IEEE 754 Double precision Floating point number using Real to Float converter module.
3. The output of real to float converter module is then fed to iteration module where the implementation of Newton Raphson iterative formula (as shown in Eqn 1) takes place.
4. The iterative module consists of Floating point Arithmethic operation modules like Addition/Substraction, Multiplication, Division, Power (Exponent) which works in a flow as shown in the Figure 3.
5. The output of first iteration i.e., x_{i+1} is fed as input i.e., x_i to the iteration block itself to carry out further iterations.
6. After all iterations the output x_{i+1} is fed to floating to real converter module which converts IEEE 754 Double precision floating point to real format
7. Then the output of floating to real converter module is given to VIO in the top module, with the help of which we can observe the output.

Results:

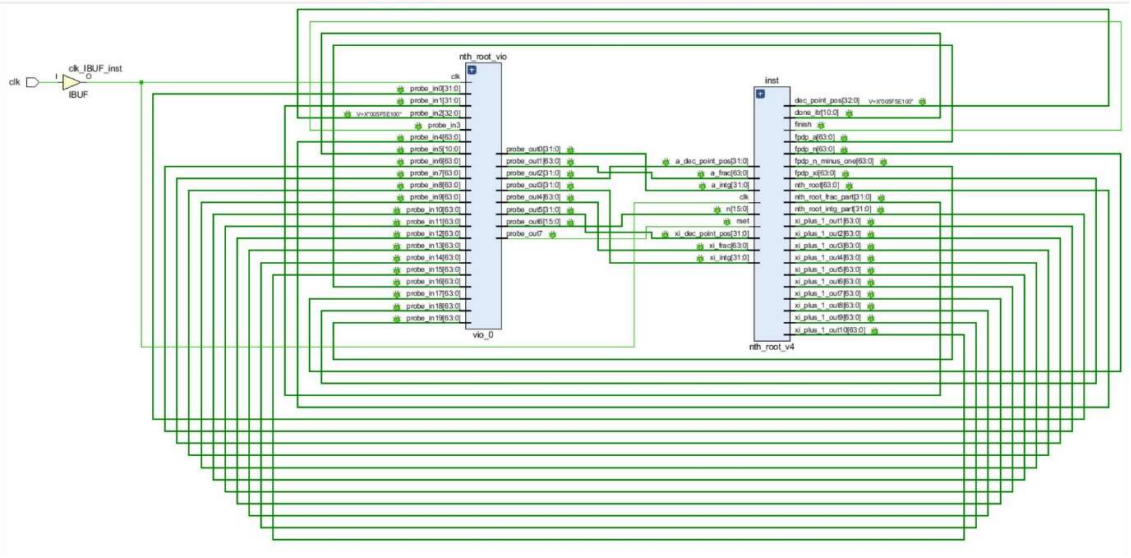

Figure 4. Final Schematic of finding n^{th} root of a number

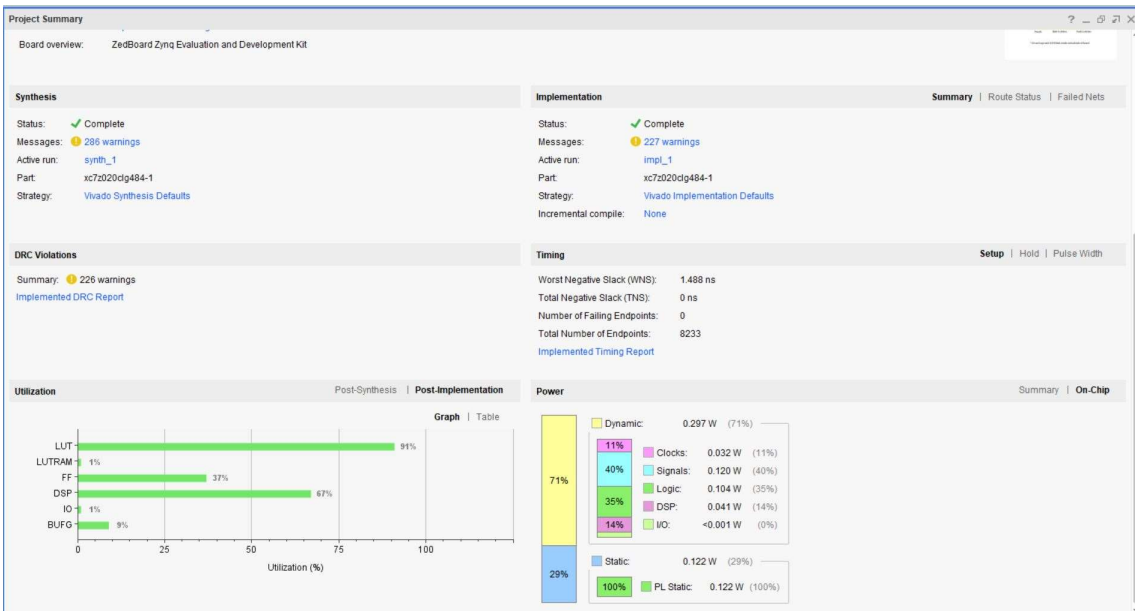

Figure 5. Project summary

From Figure 5. the implementation of finding the n^{th} root of a number using newton rapshon method has 91% LUT utilization and 67% DSP utilization. And the implementation consumes about 0.297W dynamic power and about 0.122 W static power.

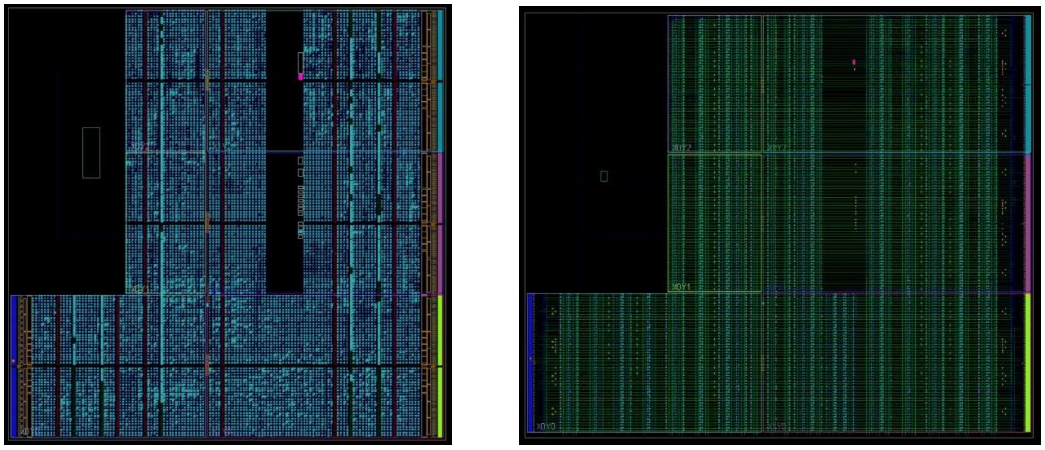

Figure 7. Device Utilization and Routing in Zynq Zedboard

Demo Output

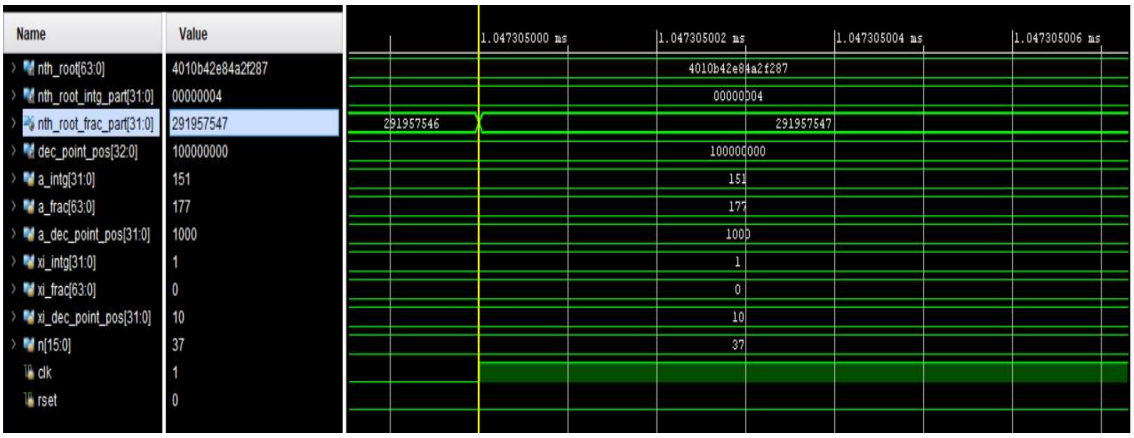

Figure 20. Post Implementation Timing Simulation

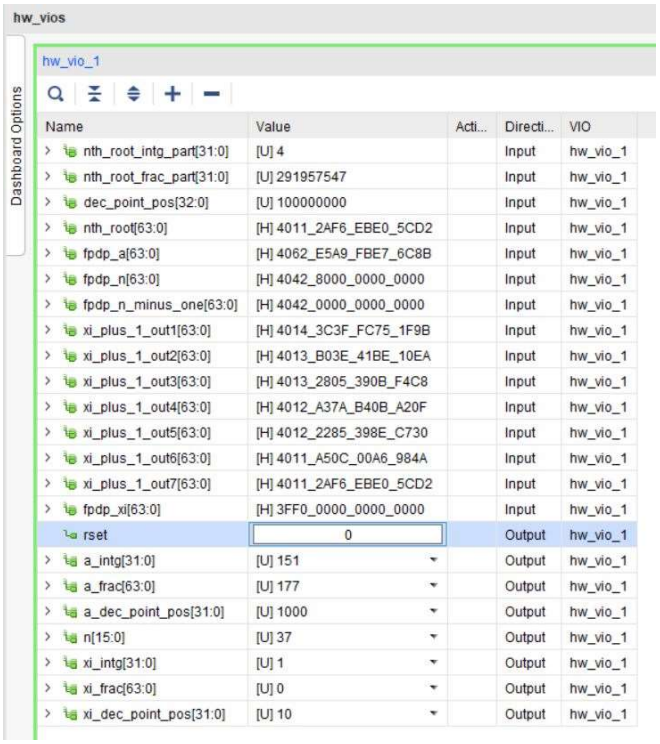

Figure 21. Demo Output Result