Implementation of Mathematical Functions using VHDL

Arushi Jain[1] Geetika Jain[2] Amita Kapoor[3]
[2]Associate Professor & Teacher Incharge & Course Coordinator
[1]Dept. of Electrical and Electronics Engg. [2]Dept. of Instrumentation and Control Engg.
[3]Department of Electronics
[1]Delhi Technological University [2]Netaji Subhas Institute of Technology
[3]Shaheed Rajguru College of Applied Sciences for Women

Abstract— The paper evaluates the use of VHDL to implement mathematical functions like Runge Kutta and Newton Raphson method. The mathematical techniques have large number of implementations ranging from applied mathematics to electrical and power systems. Analysis of these equations using VHDL is not only useful but also improves efficiency when the techniques are implemented on a large scale.

Keywords: VHDL, RTL, Counter

I. Introduction

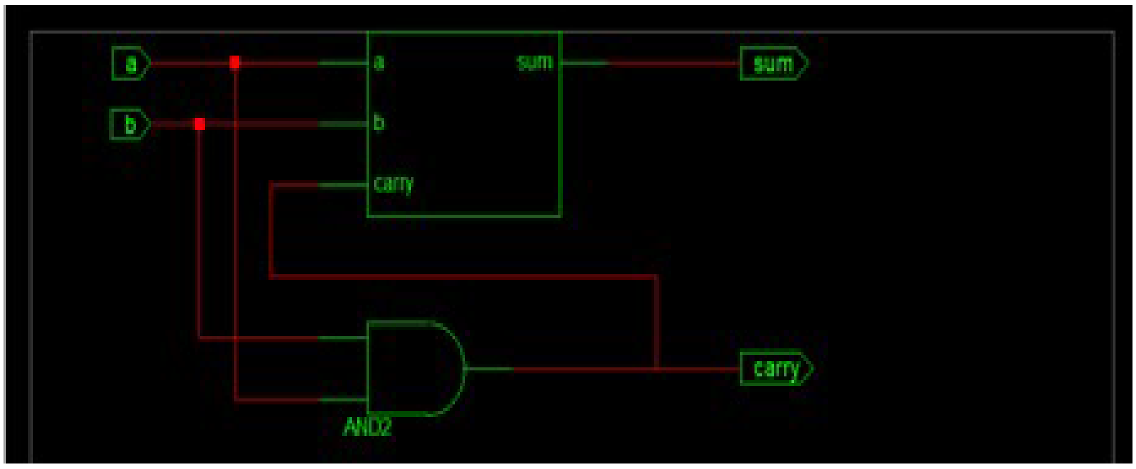
VHDL stands for Very High Speed Integrated Circuit Hardware Description Language. It is a programming language that has been designed and optimized for describing the behaviour of digital systems. One of the most important applications of VHDL is to capture the performance specification for a circuit in the form of what is commonly referred to as a test bench. In this paper, we first evaluate the implementation of basic digital circuits like adders and counters and then move on to the implementation of complex mathematical functions. These mathematical functions have a wide range of applications.

II. Adders

It is a digital circuit that performs the addition of two numbers. A half adder is a circuit which has two inputs and two outputs: a sum(S) and a carry(C). Half adder can be implemented using XOR and AND gates. In case of a full adder it has three inputs and two outputs. The third input is a carry in bit. The functional, behavioral and structural model was implemented. Their RTL schematic, test benches and the response was simulated. Fig.1 shows the RTL and Technology Schematics of a Half Adder.

III. Counter

A counter is a digital circuit which stores the number of times a particular event has occurred. It can be both clocked and non-clocked. A clocked flip flop is the one in which all the flip flops are driven from a common clock signal. Even though such hardware designs are difficult to implement they are more reliable that asynchronous type of counters. We have implemented the structural, behavioural and functional model of a modulo 10 counter. Fig.2 shows the RTL and Technology Schematics of a Counter.



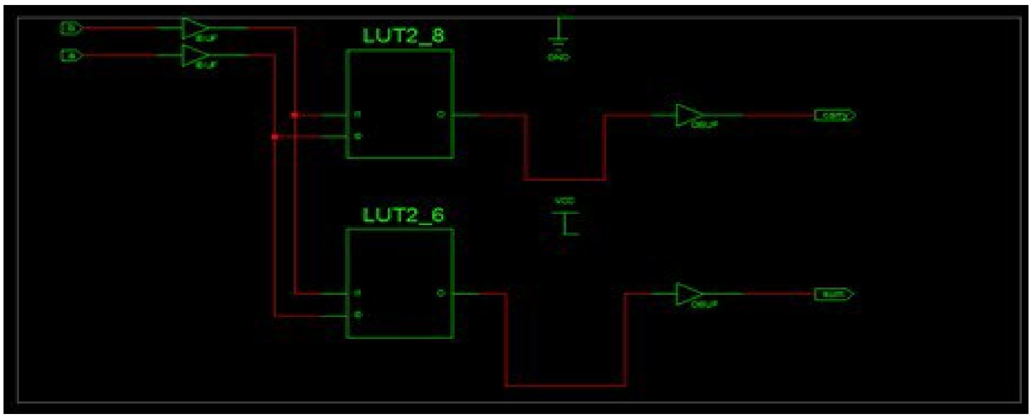


Fig. 1: RTL and Technology Schematics of a Half Adder

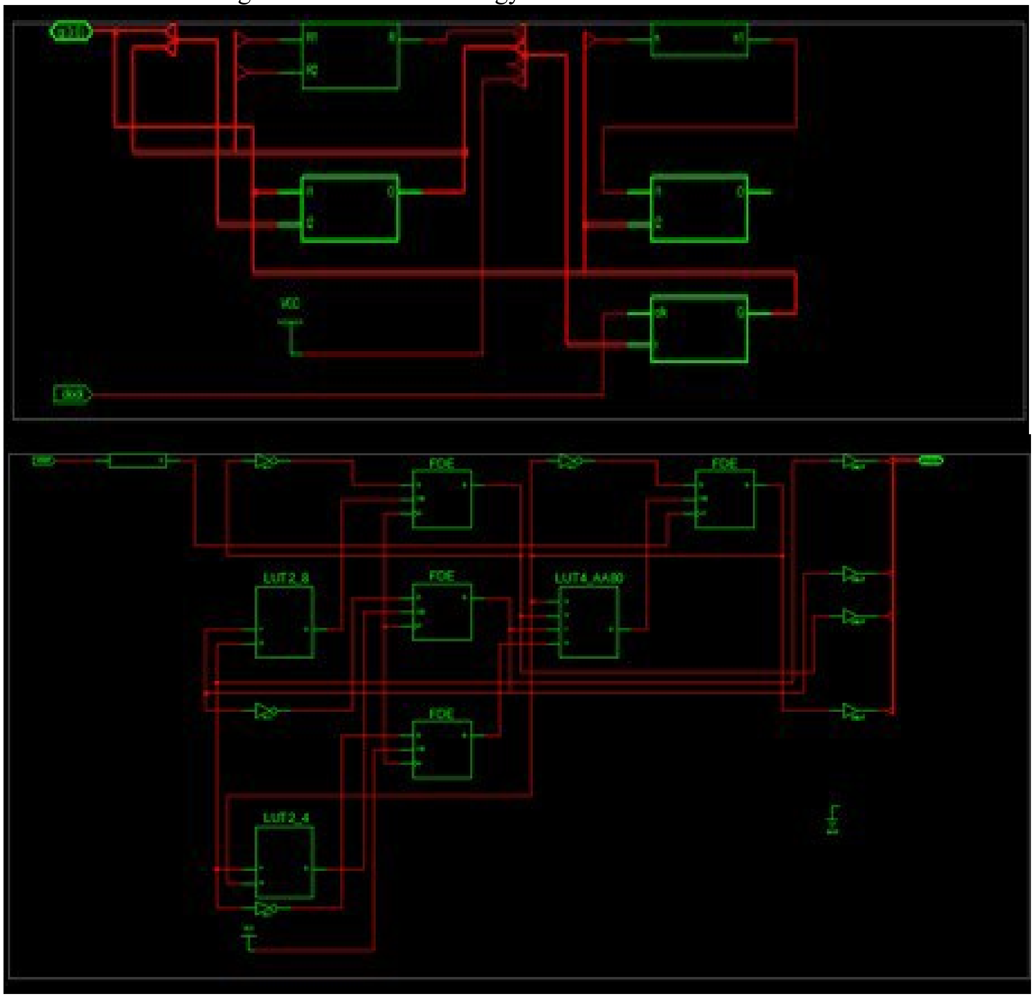


Fig. 2: RTL and Technology Schematics of a Counter.

IV. Decoder and Multiplexer

A decoder is a circuit that takes an n-digit binary number and decodes it into 2^n data lines. Decoders have large number of applications especially in microprocessor circuits where they significantly reduce the amount of hardware required.

A multiplexer is a combinational circuit designed to switch one of the several input lines through to a single common output line by the application of a control signal. It has n select lines, 2^n input lines and one output line. The behavioural and functional model of a multiplexer.

V. Runge Kutta method

Runge Kutta is a method of numerically integrating ordinary differential equations by using a trial step at the midpoint of an interval to cancel out lower order error terms. The second order formula is given by It is a single step process which involves multiple stages per step. The method was implemented using real values for the input and output variables.

$$k_1 = hf(x_n, y_n)$$

$$k_2 = hf\left(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}\right)$$

$$k_3 = hf\left(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}\right)$$

$$k_4 = hf(x_n + h, y_n + k_3)$$

$$y_(n+1) = y_n + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} + O(h^5)$$

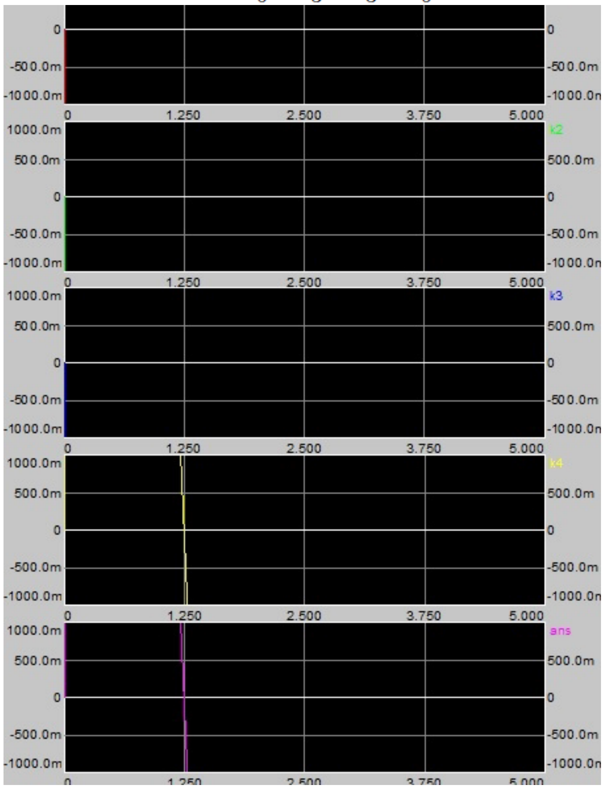


Fig. 3: Simulation results for Runge Kutta

VI. NEWTON RAPHSON METHOD

The Newton Raphson method is based on the idea of linear approximation. Let f(x) be a well-behaved function, and let r be a root of the equation f(x) = 0. We start with an estimate x_0 of r. From x(0), we produce an estimate x_1. From x_1, we produce a new estimate x_2. From x_2, we produce a new estimate x_3. We go on until we get to a value close to the value of r. This is thus an iterative method.

Let x_0 be a good estimate of r and let r = x_0 + h. Since the true root is r, and h = rx_0, the number h measures how far the estimate x_0 is from the truth. Since h is small, we can use the linear (tangent line) approximation to conclude that

$$0 = f(r) = f(x_0 + h) f(x_0) + hf_0'(x_0)$$

and therefore, unless $f'(x_0)$ is close to 0,

$$hf(x_0)/f'(x_0)$$

It follows that

$$r = x_0 + hx_0 \frac{f(x_0)}{f'(x_0)}$$

The new improved estimate x_1 of r is therefore given by

$$x_1 = x_0 \frac{f(x_0)}{f'(x_0)}$$

The next estimate x_2 is obtained from x_1 in exactly the same way as x_1 was obtained from x_0:

$$x_2 = x_1 \frac{f(x_1)}{f'(x_1)}$$

Continuing in this way. If xn is the current estimate, then the next estimate x_n + 1 is given by

$$x_n + 1 = x_n \frac{f(x_n)}{f'(x_n)}$$

VII. TRAPEZOIDAL INTEGRATION

It is a method of numerical analysis which is used for approximation of a definite integral.

$$\int_a^b f(x)\,dx.$$

The trapezoidal rule works by approximating the region under the graph of the function f(x) as a trapezoid and calculating its area. It follows that

$$\int_a^b f(x)\,dx \approx (b-a)\left[\frac{f(a)+f(b)}{2}\right]$$

Trapezoids are found as better approximations to area and hence are used.

VIII. FINITE DIFFERENTIAL TIME DOMAIN METHOD

The finite difference time-domain method is arguably the simplest, both conceptually and in terms of implementation. The disadvantage however with FDTD is that it may require a large amount of memory and computation time. The finite difference time domain method employs finite differences as approximations to both spatial and temporal derivatives that appear in Maxwell's equations. The Taylor series expansions of the function f(x) expanded about the point x_0 with an offset of $\pm\,\delta$.

$$f(x_0+\frac{\delta}{2}) = f(x_0)+\frac{\delta}{2}f'(x_0)+\frac{1}{2!}(\frac{\delta}{2})^2 f''(x_0)+\frac{1}{3!}(\frac{\delta}{2})^3 f'''(x_0)+\ldots$$

$$f(x_0-\frac{\delta}{2}) = f(x_0)-\frac{\delta}{2}f'(x_0)+\frac{1}{2!}(\frac{\delta}{2})^2 f''(x_0)-f'(x_0)+\ldots$$

Subtracting the above equations we get

$$f(x_0+\frac{\delta}{2})-f(x_0-\frac{\delta}{2}) = (\delta)f'(x_0)+\frac{2}{3!}(\frac{\delta}{2})^3 f'''(x_0)+\ldots$$

IX. CONCLUSION

VHDL can thus be used not just for implementing digital circuits but also for implementing complex mathematical functions. These functions have a wide range of applications. Newton Raphson, for instance, has applications in fluid dynamics, quantum mechanics as well as in communications. Analysing these techniques using VHDL thus makes their use in other fields to be achieved with much less difficulty.

ACKNOWLEDGMENT

We would like to express our sincere gratitude to our mentor and advisor Ms. Amita Kapoor (Professor, Shaheed Rajguru College of Applied Sciences) for her patience, motivation, enthusiasm and for enlightening us with her immense knowledge of the subject. Her cooperation and support during the whole research process was one of the main reasons we could complete our thesis with a satisfactory conclusion.

REFERENCES

[1] https://www.math.ubc.ca/ anstee/math104/104newtonmethod.pdf
[2] http://mathworld.wolfram.com/NewtonsMethod.html
[3] https://www.math.auckland.ac.nz/ butcher/CONFERENCES/JAPAN/KYUSHU/kyushuslides.pdf
[4] http://www.eecs.wsu.edu/ schneidj/ufdtd/ufdtd.pdf
[5] http://ece.wpi.edu/ wrm/Courses/EE3810/geninfo/Welcome%20to%20the%20VHDL%20Language.