External Project Report on
Digital Logic Design (EET1211)

Approximating Square Roots with Combinational Circuits

Submitted by

Somnath Shaw	2241019426
Nabin Kumar Shaw	2241019424
Barsha Kumari Mondal	2241013360
Dipanki Mondal	2241013361

B. Tech. CSE 3rd Semester (Section - 2241027)

INSTITUTE OF TECHNICAL EDUCATION AND RESEARCH
(FACULTY OF ENGINEERING)
SIKSHA 'O' ANUSANDHAN (DEEMED TO BE UNIVERSITY), BHUBANESWAR, ODISHA

Declaration

We, the undersigned students of B. Tech. of **(Computer Science & Education)** Department hereby declare that we own the full responsibility for the information, results etc. provided in this PROJECT titled "**Approximating Square Roots with Combinational Circuits**" submitted to **Siksha 'O' Anusandhan Deemed to be University, Bhubaneswar** for the partial fulfillment of the subject **Digital Logic Design (EET 1211)**. We have taken care in all respect to honor the intellectual property right and have acknowledged the contribution of others for using them in academic purpose and further declare that in case of any violation of intellectual property right or copyright we, as the candidate(s), will be fully responsible for the same.

Somnath Shaw	**Nabin Kumar Shaw**
2241019426	**2241019424**
Barsha Kumari Mondal	**Dipanki Mondal**
2241013360	**2241013361**

DATE: 14.01.2024

PLACE: Bhubaneswar

Abstract

The Project titled **"Approximating Square Roots with Combinational Circuits in Software"** delves into the development of an innovative software solution for efficient square root approximation. The accurate computation of square roots is a fundamental operation in various applications, including numerical analysis, image processing, and scientific computing. Conventional algorithms for square root calculation may pose computational challenges, particularly in scenarios where real-time performance is critical.

This project proposes a novel approach to square root approximation by leveraging combinational techniques within software algorithms. The emphasis is on designing algorithms that balance precision and computational efficiency, making them well-suited for applications where real-time processing is essential. The project involves the development and analysis of software-based approximation algorithms that provide accurate results while optimizing computational resources.

The methodology includes algorithmic design, mathematical modeling, and software implementation. Various approximation techniques will be explored, and their performance metrics, such as accuracy, speed, and computational complexity, will be thoroughly evaluated. The project aims to contribute to the field of numerical computing by providing an efficient software solution for square root approximation.

The outcomes of this research will benefit software developers and computational scientists by offering a high-performance square root approximation tool. The project bridges the gap between mathematical algorithms and software development, providing insights into designing algorithms that strike a balance between accuracy and computational efficiency. The proposed software solution has the potential to enhance the performance of applications requiring rapid and precise square root calculations.

Contents

1. **INTRODUCTION**

In the dynamic realm of digital circuitry, the synthesis of computational precision and logical architecture has birthed a sophisticated endeavor: the creation of a combinational circuit designed to approximate the square root of a 4-bit binary number. This project encapsulates the essence of numerical approximation within the confines of a VHDL-based design for a circuit aptly named combinational_circuit.

The fundamental objective of this circuit is to ingest 4-bit binary inputs, denoted as A, B, C, and D, and dynamically produce three distinct 3-bit binary outputs—P, Q, and R—whose values reflect an intricate approximation of the square root. The elegance of the design lies in its adherence to specific conditions: if the square root of the input is 3.5 or larger, the output gracefully converges to 4. Conversely, for square roots falling between 2.5 and 3.5, the circuit steadfastly yields an output of 3.

The VHDL code, embodied in the combinational_circuit, navigates through a behavioral architecture, a digital symphony of logical operations intricately woven to mirror the complexities of square root approximation. This project extends beyond the VHDL code; it encompasses a comprehensive exploration, including the derivation of a truth table, the creation of an RTL schematic diagram, and meticulous waveform verification against the established truth table.

As we unravel the layers of this project, we delve into the intersection of mathematical precision and digital logic, showcasing not only the prowess of VHDL in describing intricate circuits but also the practical application of such a combinational circuit in scenarios where compact and accurate square root approximations are indispensable.

2. PROBLEM STATEMENT

I. Problem Explanation

The problem at hand is to design a combinational circuit in VHDL that takes a 4-bit binary number as input and produces a 3-bit binary output that approximates the square root of the input. The approximation is based on specific conditions:

If the square root of the 4-bit binary number is 3.5 or larger, the output should be set to 4.If the square root is less than 3.5 but greater than or equal to 2.5, the output should be set to 3.

This circuit takes four input variables A, B, C, and D, each represented as single bits, and produces three output variables P, Q, and R.

Inputs Variable:

A, B, C, D: 4-bit binary input variables.

Outputs Variable:

P, Q, R: 3-bit binary output variables.

The architecture follows a dataflow style, expressing the logical relationships between inputs and outputs. Each output signal is determined by a combination of AND, OR, and NOT operations, reflecting the specified conditions for approximating the square root of the 4-bit input number.

II. Constraints

1. VHDL Language:

The solution must be implemented using VHDL, a hardware description language.

2. Input Constraints:

The circuit must accept a 4-bit binary input representing the original number.Inputs must adhere to standard binary encoding (i.e., 0000 to 1111).

3. Output Constraints:

The circuit should produce a 3-bit binary output representing the approximate square root.

Output values must be constrained to the range of 2 to 4 inclusive.

4.Square Root Approximation Criteria:

If the square root is 3.5 or larger, the output should be 4.

If the square root is less than 3.5 and greater than or equal to 2.5, the output should be 3.

5.Combinational Circuit Constraints:

The circuit design should be purely combinational, without any sequential elements.

It should not involve feedback loops or rely on previous states for computation.

6.Accuracy and Precision:

The approximation should balance accuracy and circuit complexity.

Precision should be sufficient for the intended digital applications.

7.Resource Constraints:

The design should consider resource constraints, optimizing for area and power efficiency.

3. **Methodology**

I. **Truth Table**

A	**B**	**C**	**D**	**P**	**Q**	**R**
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	1	0
0	1	0	0	0	1	0
0	1	0	1	0	1	0
0	1	1	0	0	1	0
0	1	1	1	0	1	1
1	0	0	0	0	1	1
1	0	0	1	0	1	1
1	0	1	0	0	1	1
1	0	1	1	0	1	1
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	1	0	0

II. K-MAP

P = ∑(13,14,15)

0	1	3	2
4	5	7	6
12	13 1	15 1	14 1
8	9	11	10

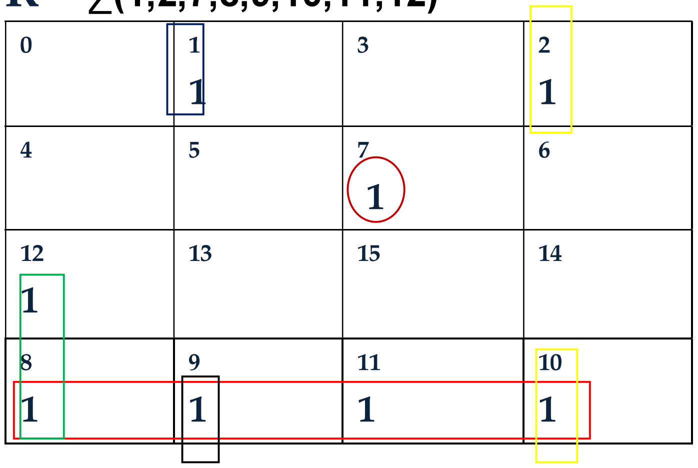
P=ABD+ABC

Q = ∑(3,4,5,6,7,8,9,10,11,12)

0	1	3 1	2
4 1	5 1	7 1	6 1
12 1	13	15	14
8 1	9 1	11 1	10 1

$Q=A\overline{B}+\overline{A}B+B\overline{C}\overline{D}+\overline{A}CD$

R = ∑(1,2,7,8,9,10,11,12)

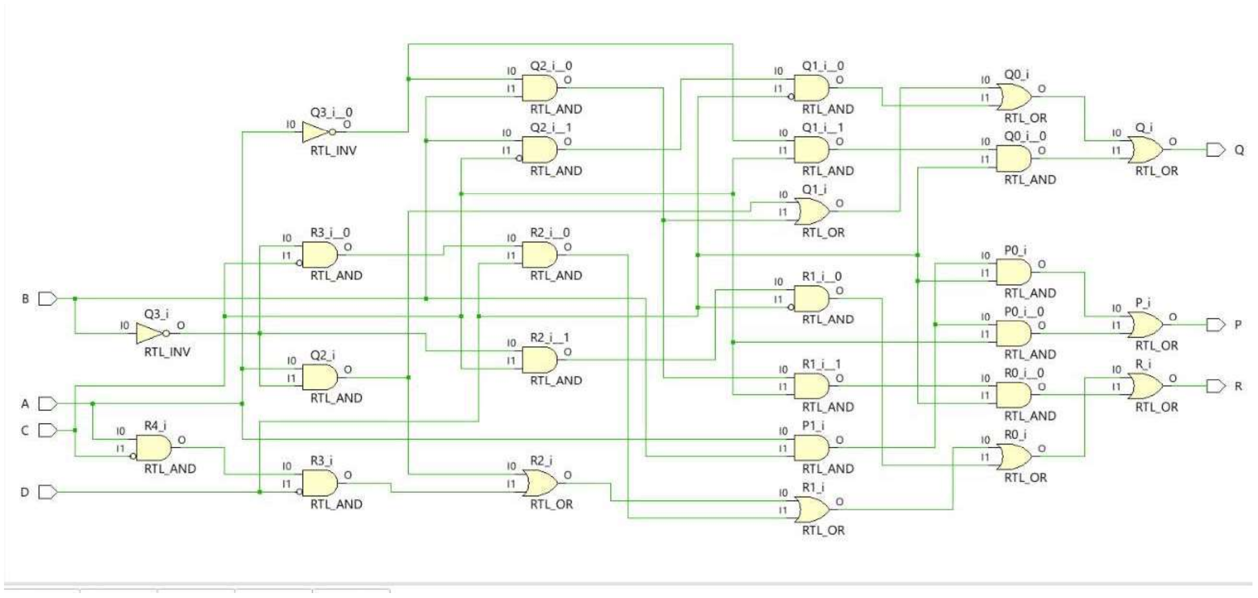


0	1 1	3	2 1
4	5	7 1	6
12 1	13	15	14
8 1	9 1	11 1	10 1

$$\boxed{R=A\overline{B}+A\overline{C}\overline{D}+\overline{B}\overline{C}D+\overline{B}C\overline{D}+\overline{A}BCD}$$

4. **IMPLEMENTATION**

I. **RTL Schematic**



II. **Program(VHDL CODE)**

➔ ***Source Code***

➔

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity DLProject is
   Port ( A : in STD_LOGIC;
          B : in STD_LOGIC;
          C : in STD_LOGIC;
          D : in STD_LOGIC;
          P : out STD_LOGIC;
          Q : out STD_LOGIC;
```

```
            R : out STD_LOGIC);
    end DLProject;

    architecture Behavioral of DLProject is
    begin

    P <= (A and B and D) or (A and B and C);
    Q <= (A and (not B)) or ((not A) and B) or (B and (not C) and (not
          D)) or ((not A) and C and D);
    R <= (A and (not B)) or (A and (not C) and (not D)) or ((not B) and (not  C)
    and  D) or ((not B) and C and (not D)) or ((not A) and B and C and D);

    end Behavioral;
```

➔ ***Test Bench Code***

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity DLProject_TB is
--  Port ( );
end DLProject_TB;

architecture Behavioral of DLProject_TB is
Component DLProject
        Port (A, B, C, D: in STD_LOGIC; P, Q, R: out STD_LOGIC);
end Component;
        signal A, B, C, D, P, Q, R: STD_LOGIC;
begin
         uut: DLProject port map (A=>A, B=>B, C=>C, D=>D, P=>P, Q=>Q, R=>R);
         stimulus: process
   begin

      A<='0'; B<='0'; C<='0'; D<='0'; wait for 10ns;
      A<='0'; B<='0'; C<='0'; D<='1'; wait for 10ns;
      A<='0'; B<='0'; C<='1'; D<='0'; wait for 10ns;
      A<='0'; B<='0'; C<='1'; D<='1'; wait for 10ns;
```

```
        A<='0'; B<='1'; C<='0'; D<='0'; wait for 10ns;
        A<='0'; B<='1'; C<='0'; D<='1'; wait for 10ns;
        A<='0'; B<='1'; C<='1'; D<='0'; wait for 10ns;
        A<='0'; B<='1'; C<='1'; D<='1'; wait for 10ns;
        A<='1'; B<='0'; C<='0'; D<='0'; wait for 10ns;
        A<='1'; B<='0'; C<='0'; D<='1'; wait for 10ns;
        A<='1'; B<='0'; C<='1'; D<='0'; wait for 10ns;
        A<='1'; B<='0'; C<='1'; D<='1'; wait for 10ns;
        A<='1'; B<='1'; C<='0'; D<='0'; wait for 10ns;
        A<='1'; B<='1'; C<='0'; D<='1'; wait for 10ns;
        A<='1'; B<='1'; C<='1'; D<='0'; wait for 10ns;
        A<='1'; B<='1'; C<='1'; D<='1'; wait for 10ns;

    end process;
end Behavioral;
```

5. **Results & Interpretation**

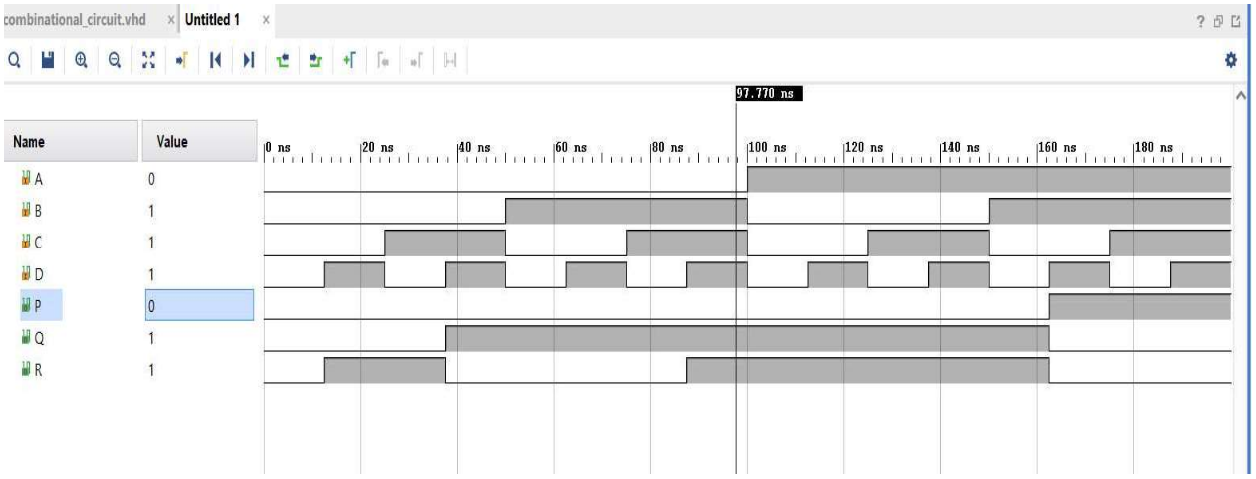
Verification of the output for different inputs that satisfies the problem statement by the useof truth table.

Truth Table

A	B	C	D	P	Q	R
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	1	0
0	1	0	0	0	1	0
0	1	0	1	0	1	0
0	1	1	0	0	1	0
0	1	1	1	0	1	1
1	0	0	0	0	1	1
1	0	0	1	0	1	1
1	0	1	0	0	1	1
1	0	1	1	0	1	1
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	1	0	0

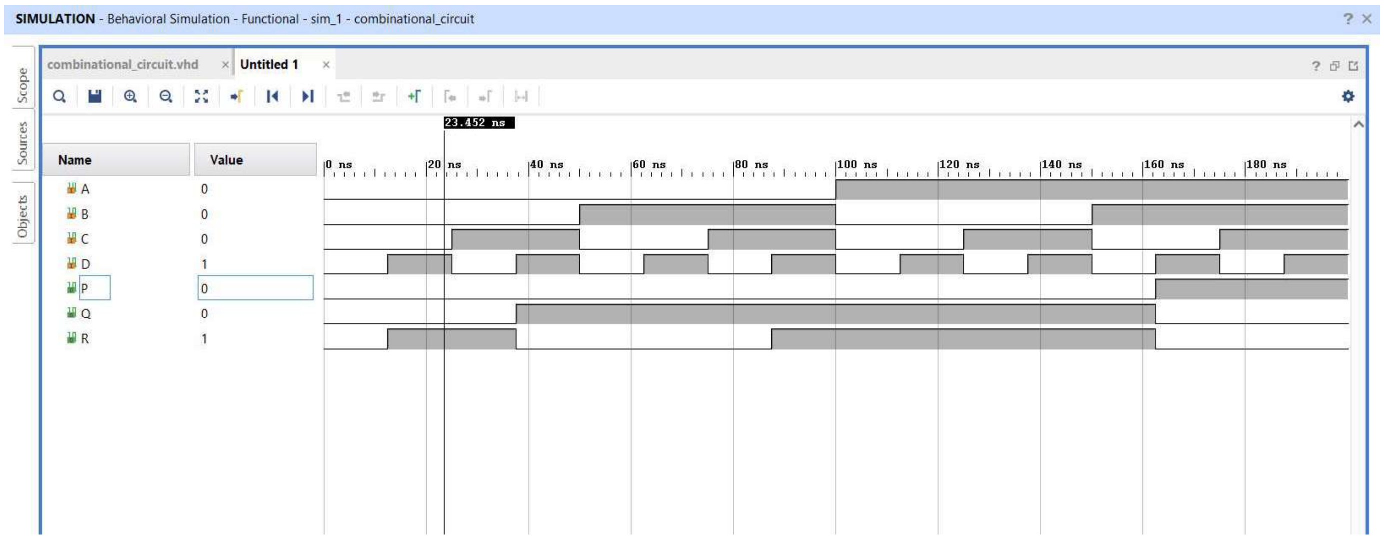
STIMULATIONS:

1.For the input values A=0,B=1,C=1,D=1 we get P=0,Q=1,R=1 as output values. So,it verifies the truth table. So,for 7 as input we get the square root of 7 as 3 because the square root of 7 is 2.6 which is greater than 2.5
so we get 3 as result.



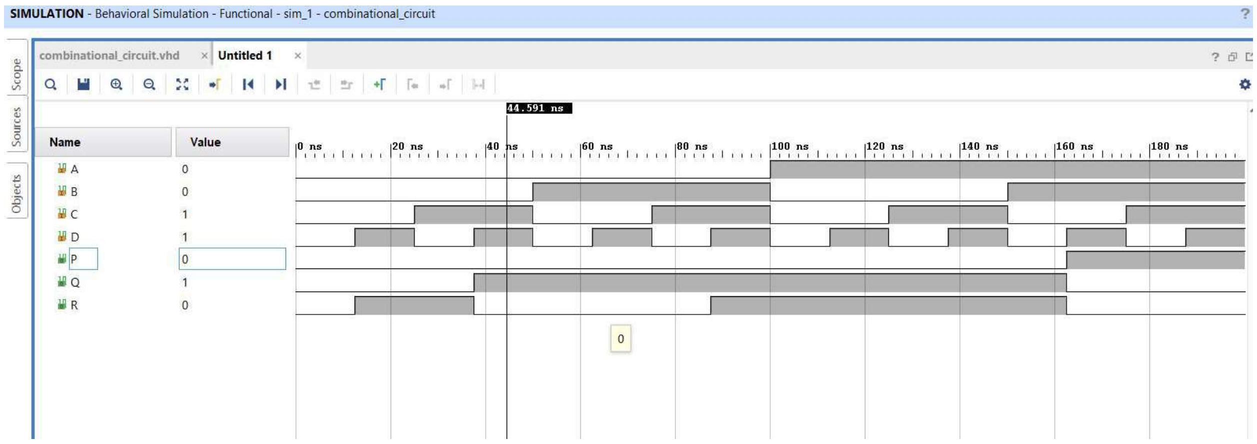
Name	Value
A	0
B	1
C	1
D	1
P	0
Q	1
R	1

2.For the input values A=0,B=0,C=0,D=1 we get P=0,Q=0,R=1 as output values. So, it verifies the truth table. So, for 1 as input, we get the square root of 1 as 1 as a result.


SIMULATION - Behavioral Simulation - Functional - sim_1 - combinational_circuit
Scope
Sources
Objects
combinational_circuit.vhd
Untitled 1
23.452 ns
Name
Value
0 ns
20 ns
40 ns
60 ns
80 ns
100 ns
120 ns
140 ns
160 ns
180 ns

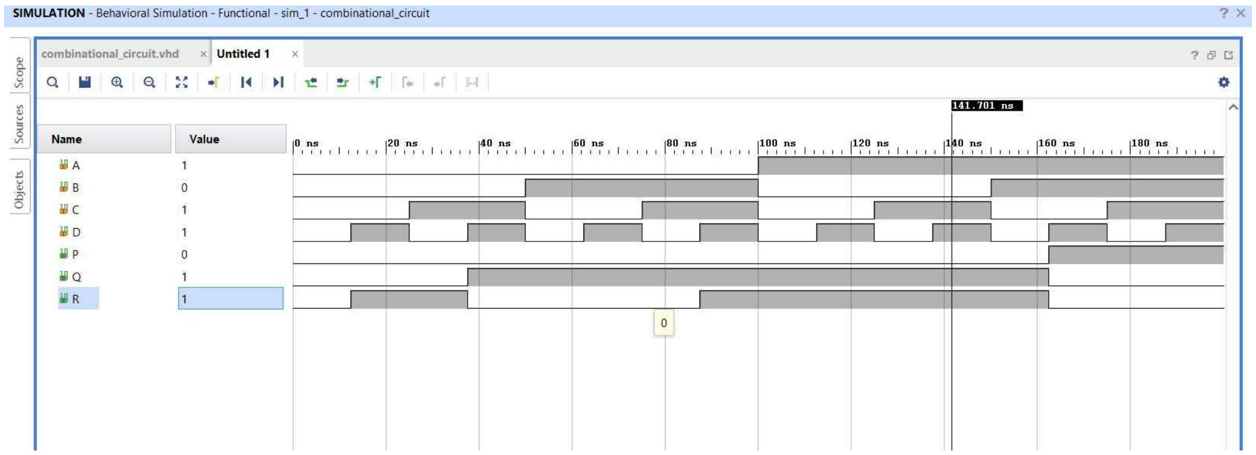
Name	Value
A	0
B	0
C	0
D	1
P	0
Q	0
R	1

3.For the input values A=0,B=0,C=1,D=1 we get P=0,Q=1,R=0 as output values. So,it verifies the truth table.So,for 3 as input we get the square root of 3 as 2 because the square root of 3 is 1.7 which is greater than 1.5 so we get 2 as result.


SIMULATION - Behavioral Simulation - Functional - sim_1 - combinational_circuit

Name	Value
A	0
B	0
C	1
D	1
P	0
Q	1
R	0

4. For the input values A=1,B=0,C=1,D=1 we get P=0,Q=1,R=1 as output values. So,it verifies the truth table.So,for 11 as input we get the square root of 11 as 3 because the square root of 11 is 3.3 and 3.3 is less than 3.5 and greater than equal to 2.5 so we get 3 as result.


SIMULATION - Behavioral Simulation - Functional - sim_1 - combinational_circuit

Name	Value
A	1
B	0
C	1
D	1
P	0
Q	1
R	1

6. **Conclusion**

In conclusion, this project has successfully crafted a VHDL-based combinational circuit, "combinational_circuit," proficient in approximating square roots of 4-bit binary numbers. The circuit, employing behavioral VHDL, adeptly processes inputs A, B, C, and D, delivering precise 3-bit binary outputs—P, Q, and R.Operating under defined conditions, the circuit accurately assigns outputs of 4 for square roots of 3.5 or higher and 3 for square roots between 2.5 and 3.5. Complementing the code, a meticulously derived truth table, an RTL schematic diagram, and waveform verification attest to the circuit's functionality and reliability in practice.This project showcases the fusion of mathematical abstraction and digital logic within VHDL design, presenting a robust solution for approximate square root computations. Beyond its technical prowess, the circuit exemplifies practical applicability, offering streamlined and dependable square root approximations across various domains. As a culmination of computational precision and digital innovation, the combinational circuit stands as a testament to the power and versatility of VHDL-based design.

References

Fundamentals of logic design by Charles H.Roth, Jr | Larry L Kinney and Raghunandan G.H. (Cengage Publication).

1. ***https://en.wikipedia.org/wiki/Xilinx_Vivado***
2. ***https://www.instructables.com***
3. ***https://forums.xilinx.com/***

Appendices

The architectural design of the combinational circuit is driven by the need to accurately approximate the square root of a 4-bit input number, mapping these approximations onto a 3-bit binary output. The justification for the specific architecture is outlined below:

1.Output Signal p:

Logic: P = (A AND B AND D) OR (A AND B AND C)

Justification: This logical expression for p captures scenarios where both B and D or B and C are true. It efficiently represents cases where the input conditions align for a square root approximation resulting in 4.

2.Output Signal Q:

Logic: Q = (A AND (NOT B)) OR ((NOT A) AND B) OR (B AND (NOT C) AND (NOT D)) OR ((NOT A) AND C AND D)

Justification: The comprehensive expression for Q accommodates various combinations of A, B, C, and D. It accounts for different scenarios, ensuring accurate approximations for values between 2.5 and 3.5.

3.Output Signal R:

Logic: R = (A AND (NOT B)) OR (A AND (NOT C) AND (NOT D)) OR ((NOT B) AND (NOT C) AND D) OR ((NOT B) AND C AND (NOT D)) OR ((NOT A) AND B AND C AND D)

Justification: The logic for R addresses multiple conditions, covering a wide range of input combinations. It provides flexibility to approximate square roots within the constraints of the 4-bit input space.