International Journal of Electronics and Electical Engineering

Volume 2 | Issue 3 Article 16

January 2014

# EARLY ESTIMATION OF DELAY IN BINARY TO BCD CONVERTOR

D. JOTHSNA NARSIMHAM
*Dept. of Electronics & Communication, SCET, Seetharampuram, India*, jothsna1987@yahoo.com

P. LAKSHMI SAROJINI
*Dept. of Electronics & Communication, SCET, Seetharampuram, India*, sagarlakshmip@gmail.com



Recommended Citation
NARSIMHAM, D. JOTHSNA and SAROJINI, P. LAKSHMI (2014) "EARLY ESTIMATION OF DELAY IN BINARY TO BCD CONVERTOR," *International Journal of Electronics and Electical Engineering*: Vol. 2 : Iss. 3 , Article 16.
DOI: 10.47893/IJEEE.2014.1102
Available at: https://www.interscience.in/ijeee/vol2/iss3/16

# EARLY ESTIMATION OF DELAY IN BINARY TO BCD CONVERTOR

**D. JOTHSNA NARSIMHAM[1] & P. LAKSHMI SAROJINI[2]**

[1,2]Dept. of Electronics & Communication, SCET, Seetharampuram, India
E-mail: jothsna1987@yahoo.com, sagarlakshmip@gmail.com

**Abstract -** A novel high speed architecture for fixed bit binary to BCD conversion which is better in terms of delay is presented in this paper. In recent years, decimal data processing applications have grown and thus there is a need to have hardware support for decimal arithmetic. Decimal digit multipliers are having Binary to BCD conversion as the basic building block. This decimal multiplication in turn is an integral part of commercial, internet and financial based applications.

***Keywords****-Binary to BCD conversion; decimal multiplication; decimal arithmetic.*

## I. INTRODUCTION

In commercial business and internet based applications, decimal arithmetic is receiving significant importance. Decimal arithmetic is important in many applications such as banking, tax calculation, insurance, accounting and many more. Even though binary arithmetic is used widely, decimal computation is essential. The decimal arithmetic is not only required when numbers are presented for inspection by humans, but also it is a necessity when fractions are being used. Rational numbers whose denominator is a power of ten are decimal fractions and most them cannot be represented by binary fractions. For example, the value 0.01 may require an infinitely recurring binary number. Even though the arithmetic is correct, but if binary approximation is used instead of an exact decimal fraction, results can be wrong.

For example, consider a calculation involving a 5% sales tax on an item such as a $0.25 call on a telephone. It is then rounded to the nearest cent. Here double precision binary floating point is used and the result of multiplying 0.25 by 1.05 is a little under 0.2624999999999999999 whereas the result would be 0.2625 by using decimal fractions.

The latter would then be rounded to 0.2625, but by using the binary fraction the result would be rounded to incorrect $0.263. Hence, for this reason, the results for financial or any calculations must be matched which those calculated by hand and this is carried out using decimal arithmetic. Also in commercial databases, the numbers are predominantly decimal numbers. A wide range of applications that were covered by these databases include insurance, financial analysis, airline systems, banking and many more.

This extensive use of decimal data indicates that it is important to analyze how the data can be used and how the decimal arithmetic can be defined. However, the current general purpose computers perform decimal computations using binary arithmetic. But the problem is that decimal numbers such as 0.3 cannot be represented precisely in binary. The errors that result on conversion between decimal and binary formats cannot be tolerated as long as precision is concerned. Since decimal arithmetic has gained wide importance in financial analysis, banking, tax calculations. Insurance, telephone billing, such errors cannot be tolerated.

This in turn can be overcome by using a decimal arithmetic and logic unit (D-ALU). The operations related to decimal arithmetic are typically slower, more complex and occupy more area and this leads to more power and less speed when implemented in hardware. Hence, enhancing speed and reducing area is the major consideration while implementing decimal arithmetic. In this paper we will be estimating the delay using Mentor Graphics Tool.

## II. NEED OF BINARY TO BCD CONVERSION

Current arithmetic units are typically binary based and not decimal based. This due to the following two reasons: (1) It is very efficient to store binary data, (2) Manipulation of binary data is very quick on digital computers.

But since, decimal arithmetic is more advantageous than binary arithmetic, the conversion of binary data to BCD data is required. Decimal multiplication is the fundamental operation for any hardware implementation of decimal arithmetic and it is also an integral part to the above mentioned decimal-dominant applications.

## II. DECIMAL MULTIPLICATION

Decimal multiplication is the fundamental operation for the hardware implementation of decimal arithmetic. However, compare to binary multiplication, the decimal multiplication is more complicated. This is due to the following two reasons: (1) A great number of multiplicand multiples is required, (2) Decimal values with two state devices cannot be represented efficiently.

Because of the above reasons, the partial product generation becomes complicated, while the partial product accumulation is complicated due to the 2nd reason. An iterative approach is usually approached for implementing decimal multipliers. The method employed is that one multiplier digit is multiplied with the entire multiplicand and partial product is generated in each cycle. This partial product is then added to an intermediate product register in which the previously accumulated partial products are stored. This method can be better understood by the following diagram.

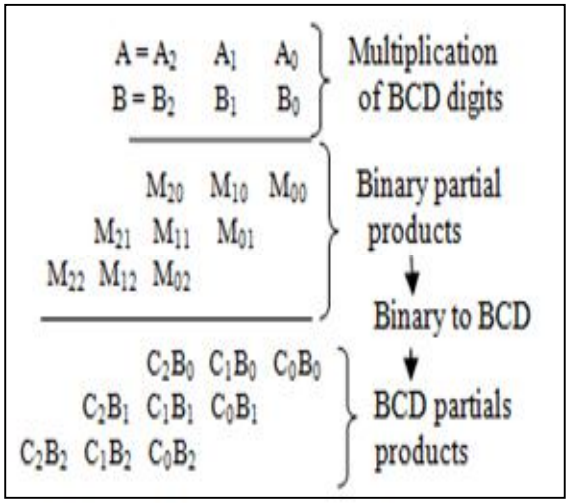



**Figure 1: Illustration of BCD conversion in BCD**

## III. ALGORITHMS

Before you begin to format your paper, first write and save the content as a separate text file. Keep your text and graphic files separate until after the text has been formatted and styled. Do not use hard tabs, and limit use of hard returns to only one return at the end of a paragraph. Do not add any kind of pagination anywhere in the paper. Do not number text heads-the template will do that for you.

Finally, complete content and organizational editing before formatting. Please take note of the following items when proofreading spelling and grammar:

### *A. Abbreviations and Acronyms*

The algorithm explained in this paper has the main objective of performing highly efficient fixed bit binary to BCD conversion considering delay to be the main criteria.

The most popularly used multipliers use 7-bit binary to 8-bit/ 2-digit BCD converters. Let the seven bits that need to be converted into 2-digit BCD digits be $a_6a_5a_4a_3a_2a_1a_0$.

The binary numbers are then split into two parts as given below in order to convert these binary bits into 2-digit BCD

- The lower significant bits (i.e., LSBs) $a_3$, $a_2$, $a_1$ and $a_0$ consists of the first part.
- The remaining higher significant bits (HSBs) $a_6$, $a_5$ and $a_4$ consist of the second part.

The BCD digit can be directly represented by the lower significant part (LSBs) which has the same weight as that of a BCD digit. This case will be violated only when $a_3a_2a_1a_0$ exceeds $(1001)_2$ and if this case occurs then $(0110)_2$ needs to be added to it. The procedure of adding $(0110)_2$ when the number becomes more than $(1001)_2$ is known as **Correction** in BCD arithmetic. In this procedure, whatever the carry is obtained is added to the higher significant BCD digit which in turn is calculated from the HSBs of the binary number. Not only the higher significant digits but also the lower significant BCD digit is contributed by the HSBs. Then after BCD correction these contributions of HSBs are added to the lower significant digit.

The resulting sum is then checked for the case $(1001)_2$ and to obtain the final lower significant BCD digit, correction is done. There are 6 combinations of $a_6$, $a_5$ and $a_4$ (HSBs) possible when two BCD digits are multiplied and these combinations can be 000, 001, 010, 011, 100 and 101. There can be different contribution of each of these combinations towards lower and higher significant BCD digits. The method of calculating these contributions is explained below.

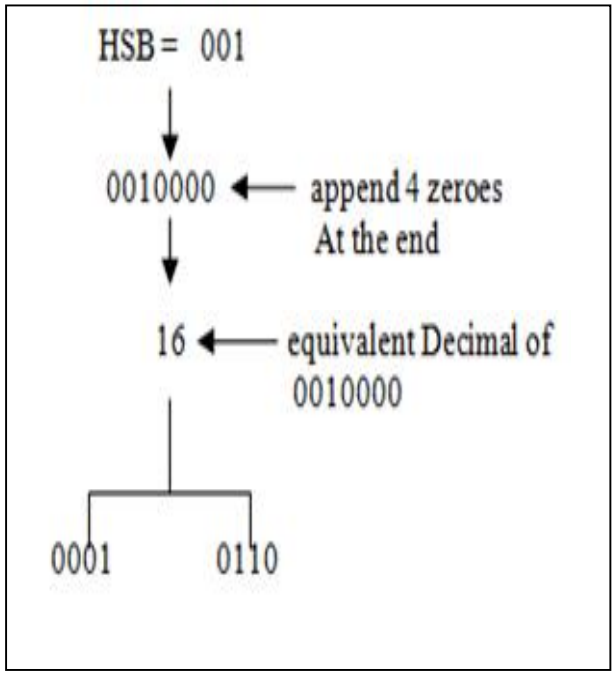



**Figure 2: Method of calculating contributions**

Thus, here 0001 corresponds to lower significant digit and 0110 corresponds to higher significant digit. Thus for all the 6 combinations, the lower and higher significant digits can be calculated. The above method is explained with an example of binary number as $(0011111)_2$.

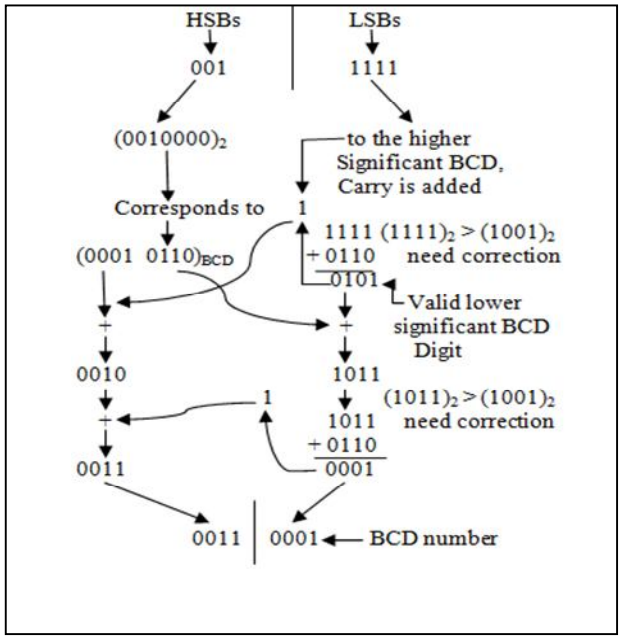



**Figure 3: Algorithm for BCD conversion.**

## IV. IMPLEMENTATION

In this section, the architecture is explained in detail. This implementation focuses on reduction in delay and area and is shown in figure below. In this implementation, $a_6a_5a_4a_3a_2a_1a_0$ are the binary bits that need to be converted to BCD bits $z_7z_6z_5z_4\ z_3z_2z_1z_0$.

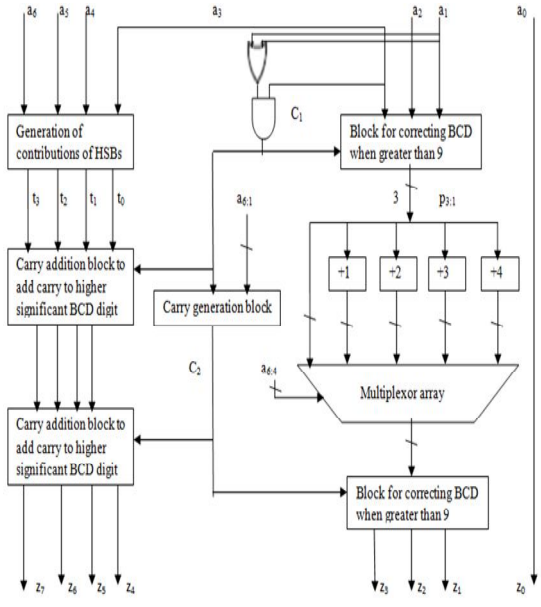



**Figure 4: Implementation of algorithm**

Here $z_0$ is the same as $a_0$ and hence no operation is done on $a_0$. In order to check whether the LSBs are greater than $(1001)_2$ or not, $a_3$, $a_2$ and $a_1$ bits are used. For this verification the following equation (a) can be used.

$C_1 = (a_2+a_1).\ a_3$ (1)

In the cases when $C_1$ is high, the block for BCD correction adds $(0110)_2$ to the input bits. Figure below is the implementation of BCD correction block.

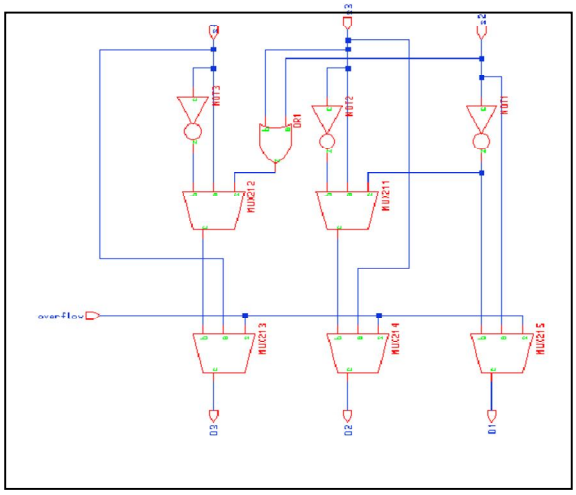


**Figure 5: BCD correction block**

In parallel, along with $a_3$, the HSBs are fed to a simple logic block called Generation of contribution block which in turn produces the higher significant. The implementation of this block shown in figure(6) below uses the equations from (b) to (e).

$t_0 = a_6' a_5' a_4 + a_5a_4'$ (2)

$t_1 = (a_6 + a_4).\ a_4'$ (3)

$t_2 = a_5 (a_3 + a_4) + a_6a_4'$ (4)

$t_3 = a_6a_4$ (5)

The carry from the lower significant digit which is $C_1$ is added to the higher significant digits $t_3t_2t_1t_0$.

This carry is added using the carry addition block which is shown in figure (7) below.

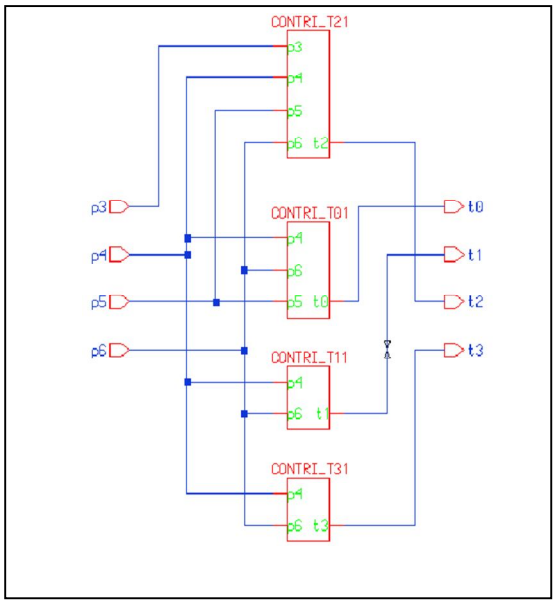



**Figure 6: Contribution generation block**

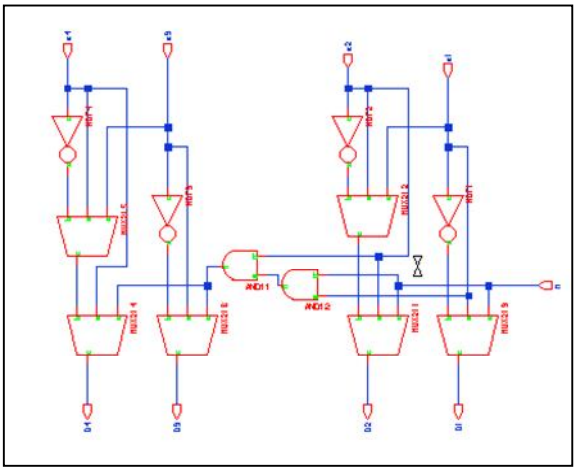
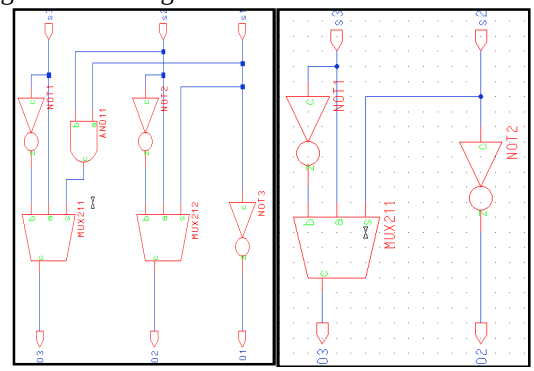
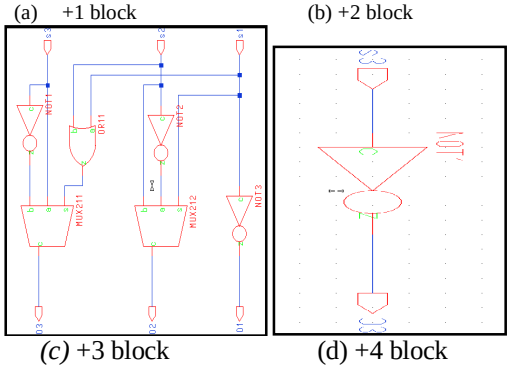


Figure 7: Carry addition block

Contribution of HSBs towards lower significant BCD digit is fixed and unique and is known immediately once HSBs are known. For this four distinct adder units have been implemented which in turn add only specified values to the inputs in parallel. These adder blocks take the correct LSBs ($p_3$, $p_2$ and $p_1$) as inputs and add specific numbers to them. These adders can be implemented through the logic shown in figures below.



(a) +1 block (b) +2 block



*(c)* +3 block (d) +4 block

Figure 8: Adder blocks

A multiplexor is then used to get the appropriate result. The selection bits of such a multiplexor are $a_6$, $a_5$ and $a_4$ i.e the HSBs. This result from the multiplexor is then fed to BCD correction block in order to check whether this result is greater than 9 or not. For this verification it makes use of carry $C_2$ as input. The bits $z_3$, $z_2$ and $z_1$ coming out of BCD correction block along with $z_0$ form the final lower significant BCD digit. The implementation of multiplexor array is shown in figure 9.

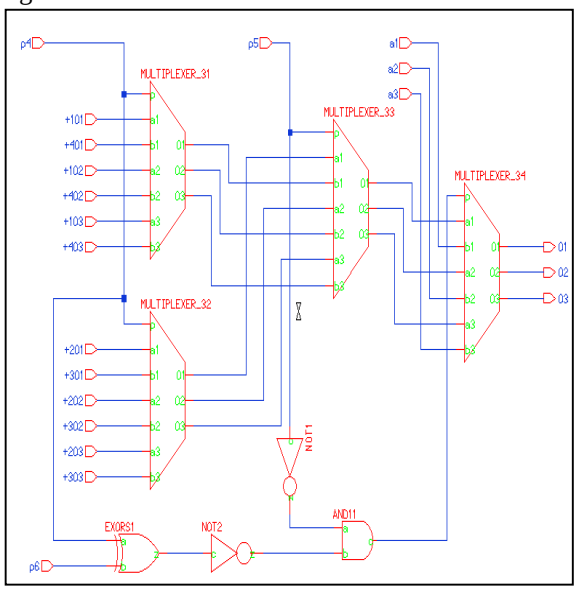


**Figure 9: Multiplexor array**

## V. LAYOUT DESIGN

This paper mainly deals with drawing the layout of the implementation mentioned above. Figures 10, 11 and 12 show the layouts of not gate and universal gates nand and nor. Since in the early stage drawing the layout of the entire implementation, the layouts of the basic gates used in the implementation has been drawn. These layouts are drawn using mentor graphics tool. First for drawing the layout, the schematic of the corresponding gate must be drawn using design architect feature of mentor graphics. After drawing schematic, using this schematic layout is drawn using ic station feature of mentor graphics. The various lambda based design rules are followed for drawing these layouts.

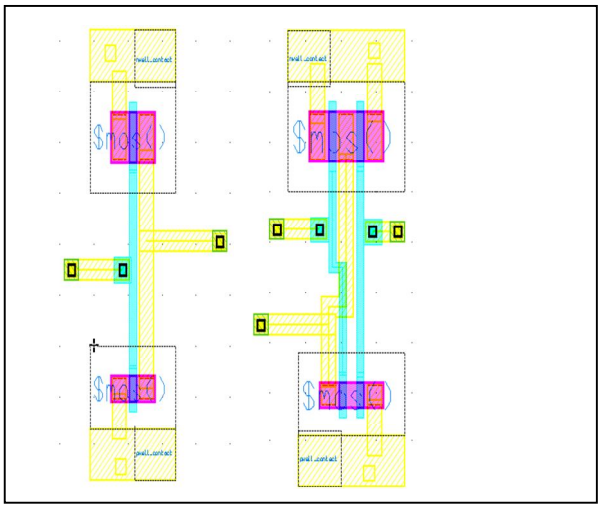


**Fig. 10: NOT gate layout Fig.11: NAND gate layout**

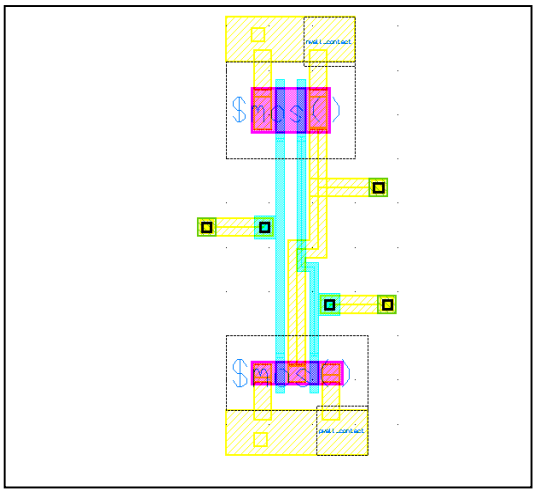


**Figure 12: NOR gate layout**

## VI. SIMULATION RESULTS

In a digital circuit, the time delay ($t_d$) mainly depends on the load capacitance ($C_L$) and the driving current. This $C_L$ in turn consists of the interconnect capacitance, the input capacitance of the driven gate and the output capacitance of the driving gate. The following are the simulation results of the various blocks of our implementation.

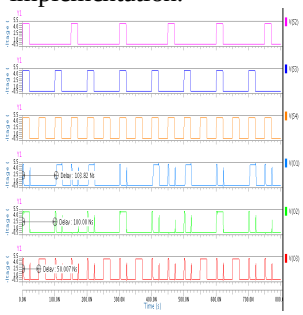


**Figure 13: BCD correction Block**

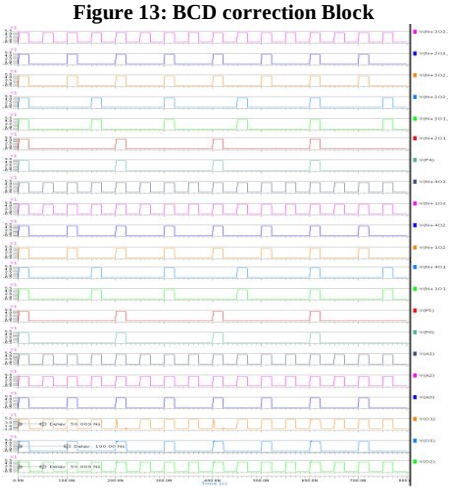


**Figure 14: Multiplexor Array**

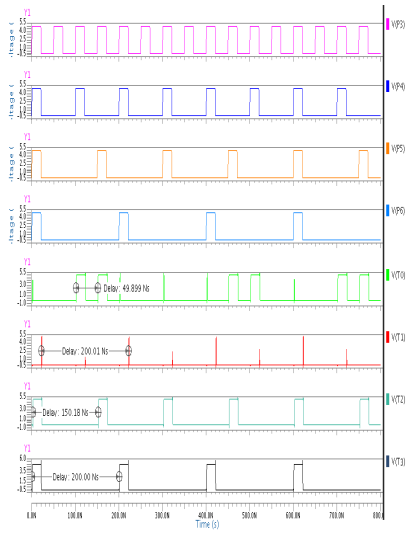


**Figure 15: Generation of Contribution Block**

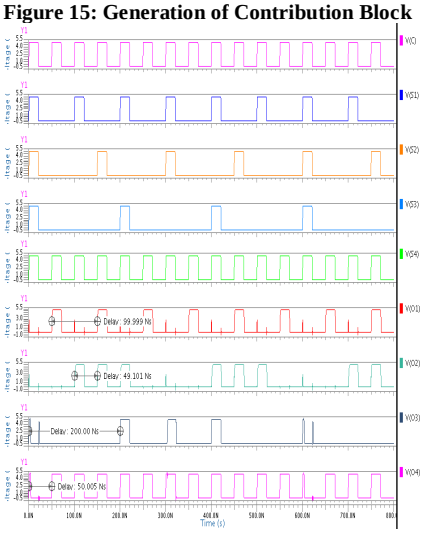


**Figure 16: Carry Addition Block**

**Figure 17: Final Binary to BCD conversion implementation**

## VII. RESULTS

Here, the delay has been calculated in order to estimate the speed of the implementation. How fast binary is converted into BCD is decided by the delay of the implementation. The table below shows the delay of various blocks in the implementation:

Table 1: Delays of various blocks

| Sr . No. | Name of the block | Delay in ns |
|---|---|---|
| 1. | BCD correction | 103.82 |
| 2. | Contribution generator | 200.01 |
| 3. | 4-bit carry addition | 99.999 |
| 4. | Carry c1 block | 100.005 |
| 5. | Carry c2 block | 250.035 |
| 6. | Multiplexor array | 50.009 |
| 7. | +1 block | 100.009 |
| 8. | +2 block | 149.999 |
| 9. | +3 block | 75.005 |
| 10. | +4 block | 25.001 |

The delay of the overall implementation is calculated as given below.

Delay= {2 * BCD correction} + {1 * contribution generation} + {2 * carry addition to HSBs} + {1 * carry c1 block} + {1* carry c2 block} + {1 * multiplexor array} + {1 * +1 block} + {1 * +2 block} + {1 * +3 block} + {1 * +4 block}

DELAY = 1357.11 ns

## REFERENCES


[1] Jairaj Bhattacharya, Aman Gupta and Anshul Singh, " A High Performance Binary to BCD Convertor For Decimal Multiplication"

[2] M. A. Erle, E. M. Schwarz and M. J. Schulte, "Decimal multiplication with efficient partial product generation," 17th IEEE Symposium on Computer Arithmetic, 2005, pp. 21-28.I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[3] M. A. Erle and M. J. Schulte, "Decimal multiplication via carry-save addition," Proceedings. IEEE International Conference on Application-Specific Systems, Architectures, and Processors, 2003, pp. 348 - 358.R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[4] A. Vazquez, E. Antelo and P. Montuschi, "A New Family of High-Performance Parallel Decimal Multipliers" 18th IEEE Symposium on Computer Arithmetic, 2007, pp. 195-204.

[5] Sreehari Veeramachaneni, M. Keerthi Krishna, L. Avinesh, P Sreekanth Reddy and M.B. Srinivas, "Novel High-Speed 16-Digit BCD Adders Conforming to IEEE 754r Format", IEEE Computer Society Annual Symposium on VLSI, 2007, pp. 343- 350

[6] R. K. James, T. K. Shahana, K. P. Jacob and S. Sasi, "Decimal multiplication using compact BCD multiplier" International Conference on Electronic Design, 2008, pp.1 – 6.

[7] G. Jaberipur and A. Kaivani, "Binary-coded decimal digit multipliers" IET Computers and Digital Techniques, Volume 1, Issue 4, July 2007, pp. 377 - 381.

Essentials of VLSI circuits and systems – Kamran Eshraghian, Eshraghian Dougles and A. Pucknell, PHI, 2005 Edition