International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue-01
January 2018

Design and Implementation of Improved 64 Bit BCD Adder with BCD multiplication

P.P.Pallavi & B K Venugopal
pallavipatne9@gmail.com[1] , bkvuvce@yahoo.com[2]
[1]PG Scholar,VLSI, University Visvesvaraya College of Engineering, Bangalore, India
[2]Associate Professor, Department of ECE, University Visvesvaraya College of Engineering, Bangalore, India

Abstract: *In the present days after increasing the complexity in the computation, internet based applications we need a fast and compact decimal adder which work with less delay and same power consumptions. A decimal digit adder is key component of any decimal hardware to support decimal arithmetic applications. Therefore, this work focuses on delivering efficient BCD digit units to be used in high performance decimal hardware accelerators. The conventional BCD adders are slow due to use of two binary adders. In this paper, we designed and implemented new high speed BCD adders which use only one binary adder. The proposed BCD adder reduces the no. of binary adders due this reduction of adders the propagation delay of BCD adder is reduced. We also implemented 64 bit BCD adder using the pipelined technique.*

Keywords: Computer arithmetic, Decimal additions, VLSI design, flagged binary adder, Correction circuit, pipeline, FPGA

I. Introduction

The binary numbering system is, by far, the most common numbering system in use in computer systems today. In days long, however, there were computer systems that were based on the decimal (base 10) numbering system rather than the binary numbering system. Such computer systems were very popular in systems targeted for business/commercial systems. Although systems designers have discovered that binary arithmetic is almost always better than decimal arithmetic for general calculations, the myth still persists that decimal arithmetic is better for money calculations than binary arithmetic. Therefore, many software systems still specify the use of decimal arithmetic in their calculations. BCD representation does offer one big advantage over binary representation: it is fairly trivial to convert between the string representation of a decimal number and its BCD representation. This feature is particularly beneficial when working with fractional values since fixed and floating point binary representations cannot exactly represent many commonly used values between zero and one (e.g., 1/10). Therefore, BCD operations can be efficient when reading from a BCD device, doing a simple arithmetic operation (e.g., a single addition) and then writing the BCD value to some other device.
While performing addition operation using BCD adder, are slow due to use of two binary adders so the propagation delay is more. This delay will affect the speed of the adder which in turn affects the speed of the entire system in which it is used. So there is a need to design the BCD adder with less delay in order to increase the speed of the operation. To further reduce power and latency in BCD addition an new BCD adder is proposed using flagged binary addition for the correction constant addition. The output of adders of first stage and flagged computation block are passed through a multiplexer. The control signal for the multiplexer is generated from a control circuit which produces 1 for sum values exceeding 9 and 0 else. But due to the use of the multiplexer the propagation delay is increased. To reduce the limitation of this BCD adder we proposed a new BCD adder**.**

II. Existing system

Normal BCD adder:

In electronic systems, BCD is an encoding for decimal numbers in which each digit is represented by its own binary sequence. It allows easy conversion to digits and results in faster calculations. When BCD numbers are added, each sum digit should be adjusted to skip the six unused codes. For instance, the addition of two decimal digits in BCD, together with a possible carry from a previous least significant pair of digits (assuming maximum value for input digits) viz., 9 + 9

International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue-01
January 2018

+1 would result in 19 .The equivalent binary sum will be in the range 0 to 19 represented in binary as 0000 to 10011and BCD as 0000 to 1 1001(the first 1 being carry and next four bits being BCD digit sum). For the binary sum equal to or less than 1001 the corresponding BCD digit is correct. However when the binary sum exceeds 1001, the result is invalid BCD digit. The addition of 6(0110)2 to the binary sum converts it to the correct digit and also produces carry. Fig.1 shows the block diagram of a 1 digit BCD adder based on the above methodology.

The input digits in binary are A3A2A1A0 and B3B2B1B0. S[3],S[2],S[1],S[0] are the outputs of the first stage 4 bit adder, to which correction bits 0110(6) is added at the second stage to produce the BCD number sum along with carry output. But the BCD adder is very simple, but also very slow due to the carry ripple effect. It also used two binary adders first to add input and second is used to add correction value in the output of first binary adder due to this reason it increases propagation delay and area.

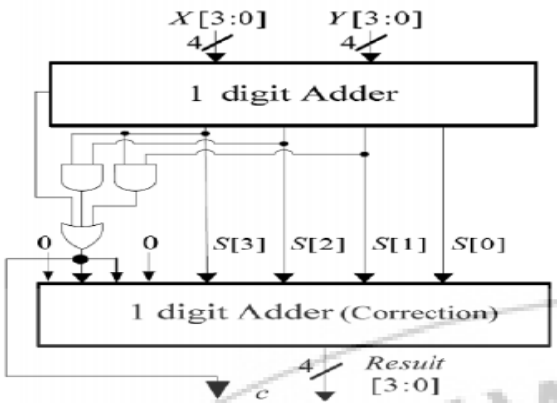


Fig 1: Normal BCD adder

Flagged BCD adder:

To reduce the limitation of normal BCD adder a new flagged BCD adder was designed. The various blocks of the proposed BCD adder are 4 bit Ripple Carry Adder(RCA), Excess 9 detector, flag bit computation block, flag inversion block and four 2:1 multiplexers whose schematic is shown in figure 2. The input A (a3a2a1a0) and B(b3b2b1b0) are fed to the first stage binary adder. The sum output S(S3S2S1S0) and carry out Co of this stage is fed to Excess 9 detector shown in figure 6(a). If the sum S(S3S2S1S0) is less than or equal to 9 the Cout of Excess 9 detector will be zero and the sum S(S3S2S1S0) will be passed out through the multiplexer. If the sum S(S3S2S1S0) exceeds 9, the Cout of Excess 9 detector will be 1 and the sum bits will be passed through the flag bit computation block to generate intermediate carry bits (d4d3d2d1) shown in equation.

$$d1=d0 \,\&\, s0$$
$$d2=d1+s1$$
$$d3=d2+s2$$
$$d4=d3+s3$$

The carry bits (d4d3d2d1) and sum S(S3S2S1S0) are then used by this block to generate flag bits (F0,F1,F2,F3).

$$F1=d1$$
$$F2=d2$$
$$F3=d3$$
$$F4=d4$$

The flag bits (F0,F1,F2,F3) and sum S(S3S2S1S0) are passed through flag inversion logic shown in fig.6(c) to generate the BCD output M3M2M1M0 for S(CoS3S2S1S0) which exceeds 9. The M3M2M1M0 of the flagged inversion block forms the other input to the multiplexer which is passed out for 1 value of Cout. The flagged BCD adder is outperformed previous designs. But due to the use of the multiplexer the propagation delay is increased. So we proposed new BCD adder use only one binary adder.

International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue-01
January 2018

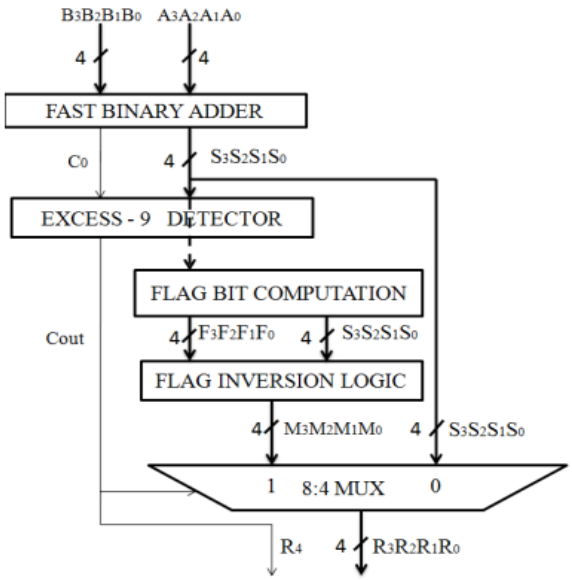


Fig2: flagged BCD adder

III. Proposed Architecture

Proposed BCD Adder

We see in flagged logic BCD adder that it uses a multiplexer in final stage which increased the propagation delay. To reduce the limitation we proposed a new architecture of BCD adder which is more efficient than conventional BCD adder in the term of speed. The architecture f proposed BCD adder is given in Figure 3.

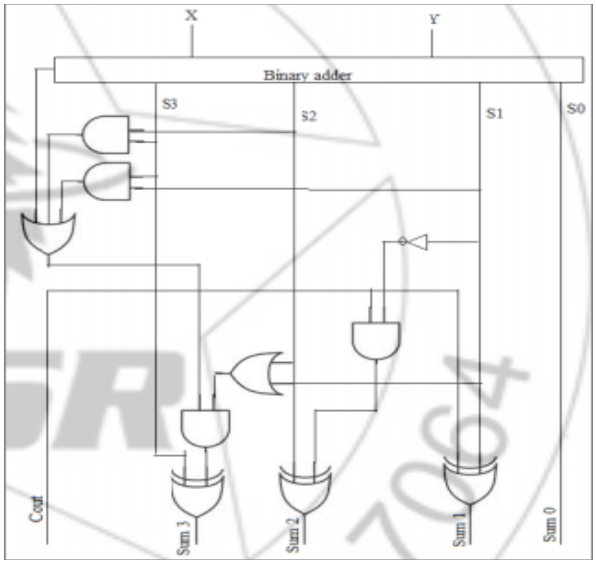


Figure 3: Proposed BCD Adders

The basic ideas for proposed new BCD adder is that in nor mal BCD adder second binary adder is use to add correction bit. The correction bits always become either 0(0000) or (0110).

When the correction bits are fix either 0 or 6 why we use second ripple adder. We can design a new logic which automatically convert binary sum in BCD sum .we can say that the correction bits add in binary s m without sing binary adder. Le t we designed the logic. Suppose the output of correction logic is cc and sum bits of first binary adder is (S (3), s (2), s (1), s (0)) Then

s(3)	s(2)	s(1)	s(0)
+ 0	cc	cc	0
Logic implementation			
sum(3)	sum(2)	sum(1)	sum(0)

Sum [0] = s [0]; ---------- -------- (1)
Sum [ ] = s [1] ^cc; ---------- ------ (2)
Sum [2] = (cc& (~s [1])) ^ s [2]; ---- (3)
sum[3]=(cc&(s[1]|s[2]))^s[3]; ------ (4)
where "^" means xor gate," ~" means not gate,| means or gate and "&" mean s and gate.

We can see in proposed BCD adder that the second binary adder is replaced my new designed logic. This logic r educes ripple effect so the propagation delay is reduced and speed of B CD adder is increased.

To reduce the ripple effect we designed a new pipelined based 64 bit BCD adders which architecture is given in figure 6. We split 64 bit BCD adder in three blocks. In first combination block there are six BCD adders to compute first six inputs. The second and third block ha s five BCD adders to compute further inputs. Now the first lock final carry signal is connected to a d flip flop which output is connected to the carry input of second block similar the third and second block connected through a d flip flop. The each block has less propagation delay then main 64 bit BCD adder. Each block operates independently.

We also analysis that in this no. of gates are also reduced like a ripple adder uses total four three input xor gates for sum an total twelve and gates for carry generation. But proposed logic uses only three two input or gates, and two and gates, on or gate and

International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue-01
January 2018

not gate. We can say it is ore efficient then previous BCD adders. Proposed 64bit pipelined BCD adder basically pipelining is a well known techniques for improving the performance of digital systems. Pipelining exploits in combinational logic in order to imp rove throughput. In this technique we split it into two or more than two separate block of combination logic. And blocks are connected with a register. E ach block ha about less delay then original block. Because each block has its own registers. all block operate independently. Working on two values at the same time. We have reduced the cycle time of the machine because we have cu t the maximum delay through the combinational logic. T his technique is used in our proposed 64 bit BCD adder. First we design 64bit BCD adder using proposed new high speed BCD adder by connecting them in series. But due the connect them in series the propagation delay is increased. It is du e to ripple effect. The architecture of this BCD adder is given in the Working on three values at the same time. The right hand block would start computing for a new input while the left ha d other block s complete t e function for the value started at the last cycle. Furthermore, we have reduced the cycle time of the machine because we have cut th e maximum delay through the small combinational logic.

64 bit BCD Adder.

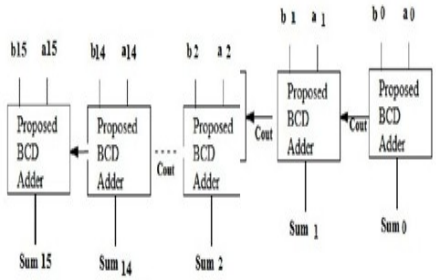


Figure 4: 64 bit BCD Adder

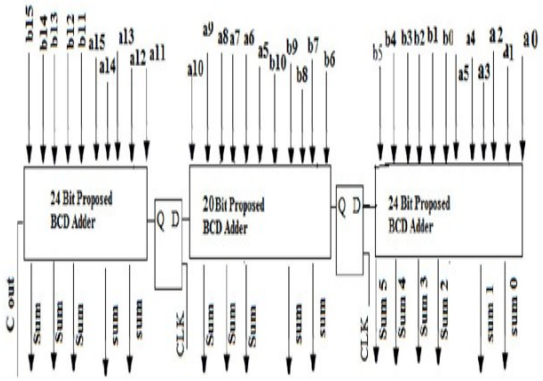


Figure 5: Proposed Pipelined 64 bit BCD Adder

EXTENSION.

BCD digit multiplication

The BCD encoding of decimal digits [0, 9] maps the latter set to [0000, 1001] such that x3x2x1x0 as the BCD encoding of X ($0 \leq X \leq 9$) satisfies the arithmetic equation 8x3 + 4x2 + 2x1 + x0 = X. A BCD-digit multiplier, with two BCD digits X and Y, realises a function p (X, Y) = X × Y, returning a product value in [0, 81] represented by two BCD digits B and C, such that p (X, Y) = 10B+C, where $0 \leq B (C) \leq 8 (9)$. The function p may be realised, in a straightforward manner, by an eight input, eight-output combinational logic or a 256 × 8 look-up table. But practical constraints on area and latency call for more optimum designs. An alternative design may use a standard 4*4 unsigned binary multiplier generating an 8-bit binary output, which should be corrected to two BCD digits, with the same arithmetic value. Given that the product value belongs to [0, 81], its most significant bit (weighted 2^7) is always zero. Let X = x3x2x1x0 and Y = y3y2 y1 y0 represent the two input BCD digits and p6p5p4 p3 p2 p1p0 is the output (i.e. product) of the standard 4*4 multiplier, with p7 = 0 ignored. Fig. 4 depicts the regular partial product generation and reduction process of this multiplier. In binary parallel multiplication, there are several techniques for partial product reduction and final product computation.

International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue-01
January 2018

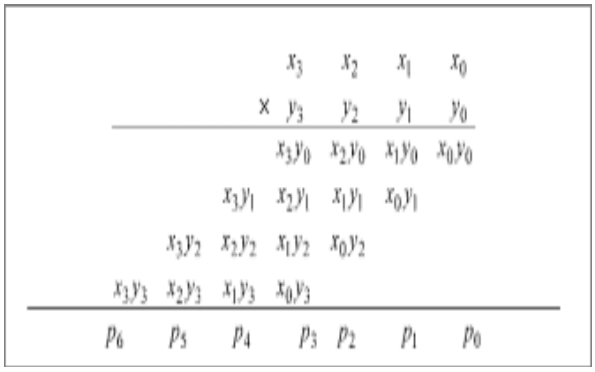
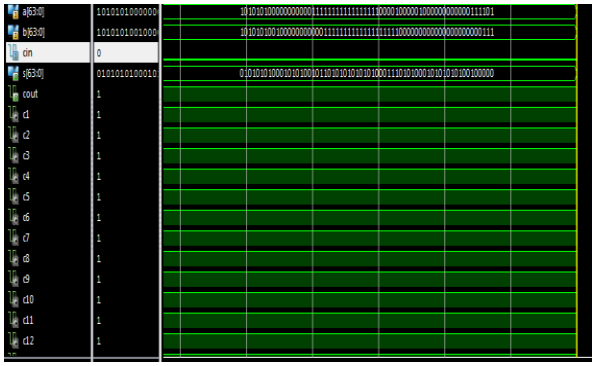


Fig. 6 BCD * BCD Binary

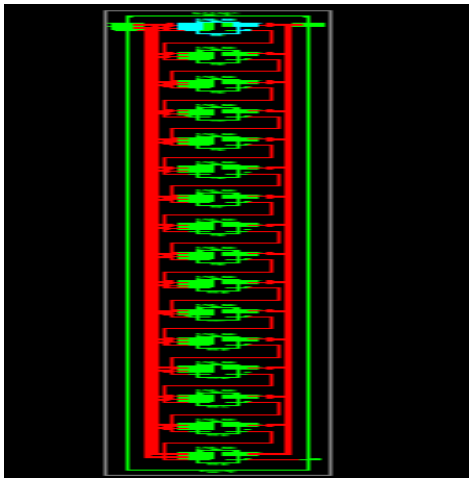
IV. Results

Existing:

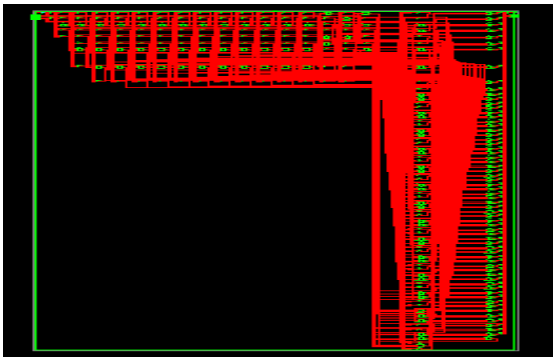
Simulation results:



RTL schematic



Technology schematic:



Design summary:

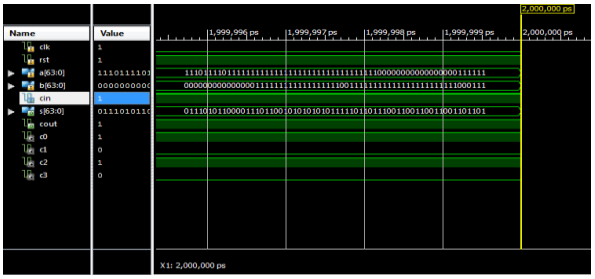
Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	71	4656	1%
Number of 4 input LUTs	128	9312	1%
Number of bonded IOBs	194	190	102%

Timing report:

```
Timing Detail:
--------------
All values displayed in nanoseconds (ns)

=========================================================================
Timing constraint: Default path analysis
  Total number of paths / destination ports: 1635775089 / 65
-------------------------------------------------------------------------
Delay:               61.147ns (Levels of Logic = 52)
  Source:            cin (PAD)
  Destination:       s<62> (PAD)

  Data Path: cin to s<62>
```

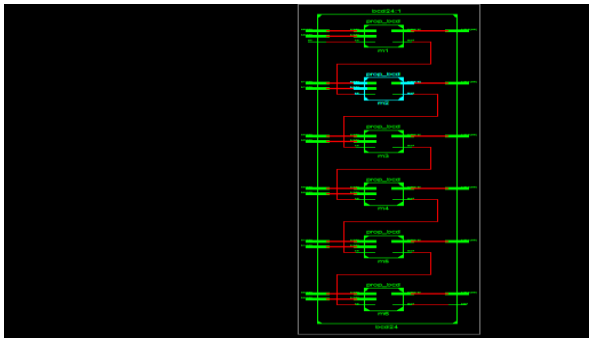
Proposed.

BCD ADDER Simulation.

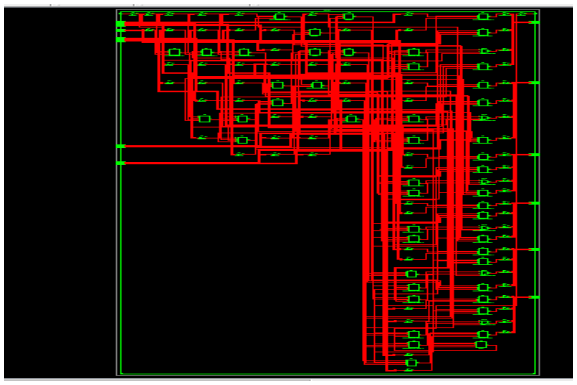


View RTL Schematic

International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue-01
January 2018



View Technology Schematic



Design summary:

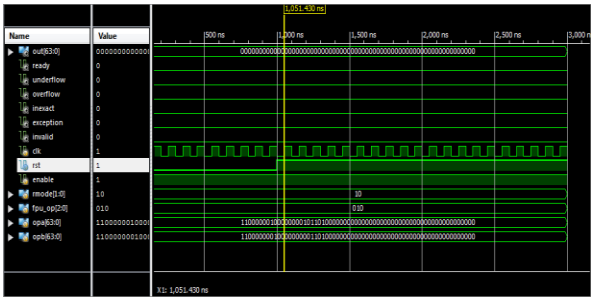
Device Utilization Summary (estimated values)			[-]
Logic Utilization	Used	Available	Utilization
Number of Slices	27	4656	0%
Number of 4 input LUTs	48	9312	0%
Number of bonded IOBs	74	232	31%

Timing Summary

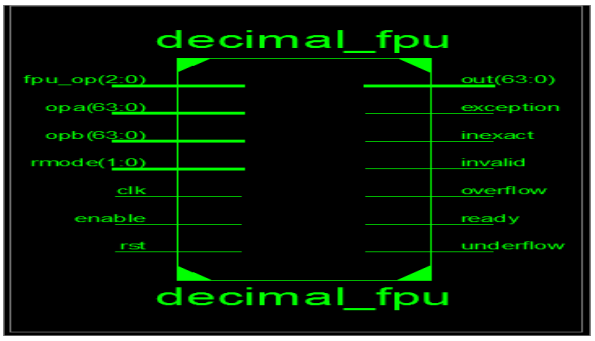
```
Timing Detail:
--------------
All values displayed in nanoseconds (ns)

================================================================================
Timing constraint: Default path analysis
  Total number of paths / destination ports: 27563 / 25
--------------------------------------------------------------------------------
Delay:               26.681ns (Levels of Logic = 22)
  Source:            cin (PAD)
  Destination:       sum5<2> (PAD)
```

Extension.

BCD MultiplierSimulation.



RTL Schematic.



Design Summary.

Device Utilization Summary (estimated values)			[-]
Logic Utilization	Used	Available	Utilization
Number of Slices	5857	4656	125%
Number of Slice Flip Flops	5377	9312	57%
Number of 4 input LUTs	10038	9312	107%
Number of bonded IOBs	206	232	88%
Number of MULT18X18SIOs	9	20	45%
Number of GCLKs	1	24	4%

Timing Summary.

```
================================================================================
Timing constraint: Default OFFSET OUT AFTER for Clock 'clk'
  Total number of paths / destination ports: 70 / 70
--------------------------------------------------------------------------------
Offset:              4.040ns (Levels of Logic = 1)
  Source:            exception (FF)
  Destination:       exception (PAD)
  Source Clock:      clk rising

  Data Path: exception to exception
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     FDRE:C->Q             1   0.514   0.357  exception (exception_OBUF)
     OBUF:I->O                 3.169          exception_OBUF (exception)
    ----------------------------------------
    Total                      4.040ns (3.683ns logic, 0.357ns route)
                                       (91.2% logic, 8.8% route)

================================================================================
```

V. CONCLUSION

In this paper, the conventional and modified proposed BCD adders are designed using Verilog. The delay of modified BCD adders is less as compared to the conventional BCD adders. We use a new logic to add the correction bits in binary sum which is faster than conventional adder .it increase the speed of BCD adder. Pipeline technique also reduces the propagation delay by increasing throughout. When implemented

on FPGA, the result proved that the proposed booth BCD adder is 15.28% faster than conventional normal BCD adder and pipeline based 64 bit BCD adder is 55.39 % faster than conventional 64 bit BCD adder. The proposed approach also applies with minor modifications to three input decimal addition.

Future Scope.

The future scope of the paper is that we can designed for three input or further inputs which will be faster than conventional decimal adders . We can also use state machine approach to increase the speed. For power reduction we can use clock gating technique.

REFERENCES

[1] Tso-Bing Juang,Hsin-Hao Peng,Chao-Tsung Kuo."Area efficient BCD Adder,"19th IEEE/IFIP International Conference on VLSI Design,2011

[2] Draft IEEE Standard for Floating-Point Arithmetic. New York: IEEE, Inc., 2004,

[3] Alp Arslan Bayrakci and Ahmet Akkas. Reduced Delay BCD Adder. IEEE, 2007.

[4] T. Lang, and A. Nannarelli, "Division Unit for Binary Integer Decimals," Prof. 20th IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP), pp. 1-7, 2009.

[5] T. Lang and A. Nannarelli, "A Radix-10 Digit-Recurrence Division Unit: Algorithm and Architecture," IEEE Transactions on Computers,

[6] Vol. 56, No. 6, pp. 727-739, June 2007. [7] R. K. James, T. K. Shahana, K. P. Jacob, and S. Sasi, "Decimal multiplication using compact BCD multiplier," Proc. International Conference on Electronic Design (ICED), pp. 1-6, 2008

[8] P. M. Kogge and H. S. Stone. A Parallel Algorithm for The Efficient Solution of a General Class of Recurrence Equations.IEEE Trans. on Computers, C-22(8), Aug. 1973.

[9] M. M. Mano. Digital Design, pages 129–131. Prentice Hall, third edition, 2002.

[10] M. S. Schmookler and A.W.Weinberger. High Speed Decimal Addition. IEEE Transactions on Computers, C-20:862– 867, Aug.1971.

[11] B. Shirazi, D. Y. Y. Young, and C. N. Zhang. RBCD: Redundant Binary Coded Decimal Adder. In IEEE Proceedings, Part E, No. 2, volume 136, pages 156–160, March 1989.

[12] J. D. Thompson, N. Karra, and M. J. SchulteB. A 64-Bit Decimal Floating-Point Adder. In Proceedings of the IEEE Computer Society Annual Symposium on VLSI, pages 297– 298, February 2004.

[13] +B. Prashanthi kumari, G. N. V. Ratna Kishor, “New Approach for Implementing BCD Adder Using Flagged Logic” international Journal of Engineering Research & Technology IJERT) Vol. 2 Issue 12, December – 2013.