International journal of basic and applied research
www.pragatipublication.com
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

# A CVNS-Based Low-Power 64-Bit Adder: Design and Implementation

**Mr. N. Chandra sekhar, Mr. T. Gangadhar Rao, Mr. J. Kiran Chandrasekhar**

**Assistant Professor[1,2], Associate Professor**

**Department of Electronics & Communication Engineering,**

**Rajamahendri Institute of Engineering & Technology, Rajamahendravaram.**

***Abstract*—** This project showcases the creation of a 64-bit mixed-signal adder that is based on the CVNS. Four 16-bit Radix-2 CVNS adders are cascaded to produce the 64-bit adder. The system's interconnections were reduced using Truncated Summation of the CVNS digits, leading to a decrease in design complexity, power consumption, and hardware costs. For use in media signal processing, this adder can execute one 64-bit, two 32-bit, and four 16-bit adds as needed. The CVNS adder is well-suited for use in multimedia applications due to its small size and low power consumption. Because of truncation summation, this system employs an algorithm for digital systems that reduces the amount of links needed. The 64-Bit CVNS adder that was synthesized using Cadence RTL Encounter has a core area of 3995 µm2, a power consumption of around 98.55 fW, and a timing slack of 7ps.It is an abstract.

***Index Terms*—** A 64-bit adder, mixed-signal adder, media signal processing, analog digits, continuous valued number system (CVNS), and computer mathematics.

**Introduction** Many different kinds of digital systems rely on the Adders in some way. One of the most important arithmetic functions for modern digital systems, fast addition has a significant influence on digital systems' overall performance. It is still difficult to add quickly while consuming little room and power, even though several adder structures, such serial and parallel structures, can perform addition. Modern central processing units (CPUs) employ adders for calculating the physical address and for all arithmetic operations. When a fully functional central processing unit (CPU) is unnecessary, adders are used in a variety of digital systems, including telecommunications systems. A wide variety of adders are known. Though ripple adders are more compact, the design calculation is painfully sluggish. Compared to ripple or carry-skip adders, carry-select adders are much faster, but they are also significantly bigger and use a lot more power. The proliferation of mobile phones, digital cameras, and other video devices has put multimedia signal processing in the spotlight [1]. For these kinds of applications, effective signal processing units need reconfigurable adders that can handle data of different lengths without increasing design complexity excessively. As a general rule, an efficient adder design may add one 64-bit, two 32-bit, four 16-bit, and eight 8-bit operations [2], which is crucial for the creation of reconfigurable systems. Implementation costs, measured in terms of worst-case latency and power consumption, tend to rise when introducing reconfigurability to an adder [3].New to computer mathematics is the CVNS representation, which stands for continuous valued number system. The development of efficient and high-performance arithmetic units, like as adders, has made good use of this continuous number system with non-integer digits.

I.

## II. CONTINUOUS VALUED NUMBER SYSTEM

CVNS [4] stands for “Continuous Valued Number System”. CVNS is a novel continuous (analog) digit representation and arithmetic system. This number system performs arithmetic operations by applying digit-level modular reduction operation on continuous real values. Some of the important and known features of the CVNS are given. These are the general arithmetic features of the CVNS, and do not consider actual system design issues of arithmetic units based on this number system. These features can be obtained by the mathematical expressions for a feasible design of a reconfigurable adder.

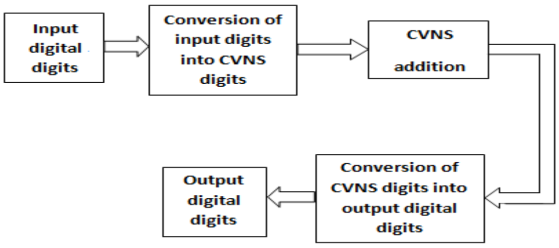

Input digital digits
Conversion of input digits into CVNS digits
CVNS addition
Conversion of CVNS digits into output digital digits
Output digital digits

Fig. 1 Block Diagram of CVNS

### A. CVNS Digits

Any value within the boundary such as $|X| \leq M$ from a positional number system with radix-B can be mapped to a set of CVNS digits to a set of CVNS digits in radix-β. The CVNS values, ((x)) are an group of CVNS digits, and can be written as a vector as follows. Where (-k ≤ i ≤ n)

$$((x)) => \{((x))_n, ((x))_{n-1}, \dots, ((x))_0 | ((x))_{-1}, \dots, ((x))_{-k}\} \quad (1)$$

The main characteristic of the CVNS digits is that they do not have a grid and can take continuous values. The CVNS digits are obtained by applying a basic modular reduction operation, in parallel, as follows:

$$((x))_i = \left( \frac{x}{M} . \beta^{n-i+1} \right) mod\beta \qquad (2)$$

Where modβ is the modulo operation on any real value and (a) modβ=a+I.β where I is an integer and M is the maximum range of representation.

Each CVNS digit consists of two parts an integer and a non-integer part which overlaps with less informed digits. The relation between any two adjacent CVNS digits is given by

$$. \ ((x))_i = \lfloor ((x))_i \rfloor + \frac{((x))_{i-1}}{\beta} \qquad (3)$$

Where $\lfloor . \rfloor$ represents floor function and $\lfloor ((x))_i \rfloor = 0,1, \dots, \beta - 1$ is an associated integer of $((x))_i$ and $\frac{((x))_{i-1}}{\beta}$ is the non integer part of CVNS digit.

### B. CVNS Addition

Addition in the CVNS is by summation of the digits without intercommunication. There are no carries in the accepted sense in the CVNS theory, and the circuitry associated with the digit generation and the manipulation shares the information at the digit level.

Considering two values, x and y, where x, y $< |M|$ digit wise CVNS addition is as follows:

$$((z))_i = ((x + y))_i$$
$$= \left( ((x))_i + ((y))_i \right) mod\beta \qquad (4)$$

The modular reduction operation in (4.4) ensures that the CVNS digits of the summation are always within the allowable range of [0,$\beta$). Therefore, if an overflow occurs in the lower informed digits, the more informed digit is not affected. The overflow is embedded within the CVNS digits.

*Example:* CVNS addition of two arbitrary values x = 58.34 and y = 72.89 shown in table 1. Maximum range of representation is considered M = 100, n = 2, and k = 2.

TABLE 1
CVNS Addition between two arbitrary values

| $i$ | 2 | 1 | 0 | -1 | -2 |
|---|---|---|---|---|---|
| $((x))_i$ | 0.5834 | 5.834 | 8.34 | 3.4 | 4 |
| $((y))_i$ | 0.7289 | 7.289 | 2.89 | 8.9 | 9 |
| $((z))_i$ | 1.3123 | 3.123 | 1.23 | 2.3 | 3 |

The CVNS Digits of ((z)) are {1.3123, 3.123, 1.23, 2.3, 3}, which is equivalent to z = 131.23 = 58.34 + 72.89.

## III. CONTROL SIGNALS

Very large scale integration analog circuits apply CVNS. A regular architecture has been used throughout the 64-bit adder to save the design time. Each of the four 16-bit adders that make up the 64-bit reconfigurable adder is further split into four uniform blocks. The system uses binary notation for both its inputs and its outputs. The system does four 16-bit additions in parallel while running in normal mode. Table 2 shows the two signals (part1 and part2) required to modify the 64-bit adder's mode of operation.. TABLE 2

Adder operation for different word lengths controlled by part1 and part2 signals

| *Part1* | *Part2* | Adder configuration |
|---|---|---|
| 0 | 0 | Byte(8-bit) |
| 0 | 1 | Half-Word(16-bit) |
| 1 | 0 | Word(32-bit) |
| 1 | 1 | Double word(64-bit) |

To partition the 16 - bit adder whenever 8 - bit operation is required, control signal breaks down the size of each of the 16 - bit adders to 8 - bits. This mode of operation is controlled by a signal denoted as ctrl8 which is generated as follows.

*Ctrl 8 ≡ (part 1 V part 2)* (5)

Each two of the 16 bit adders are combined to perform 32 - bit addition. The information is exchanged between the two adders, from the less informed adder to the more informed adder, not only in the 32 - bit but also in the 64 - bit mode of operation. From the table 2 by examining, ctrl32 signal is equivalent to

*Ctrl 32 ≡ (part 2' V part 1)* (6)

All four 16 - bit adders are combined to work as a 64 - bit adder when both part1 and part2 signals are one. Therefore, control signals for these configurations are setup as follows:

*Ctrl 64 ≡ (part 1 Λ part 2)* (7)

These control signals are used to adjust the size and resolution of the adder on-demand.

## IV. MATHEMATICAL ANALYSIS

### *A. Radix-2 16-Bit Addition*

The operations of 16-bit CVNS adder operations are converting binary data to the CVNS, adding the CVNS digits, and converting final results back to binary. The maximum range of CVNS and positional number system is as follows:
:

$$B^{m+1} = \beta^{n+1} = M \qquad (8)$$

Here, 'm' is the max index value in CVNS representation. 'n' is the max index value in positional number system.

Input digits into each of the 16-bit CVNS adders, x $_{i+t}$, represent an integer value, x $^{t}$, as follows:

$$x^t = \sum_{i=0}^{15} x_{i+t} . 2^i \qquad (9)$$

Where, t= 0,16,32,48.

By placing the previous term in (2), a direct relation between the binary and the corresponding CVNS digits for each of 16-bit CVNS adders is obtained as follows:

$$((x))_{j+t} = \left(\frac{x^t}{2^{16.}} . 2^{16-j}\right) mod2$$

$$= \left(\sum_{i=0}^{15} x_{i+t} . 2^{i-j}\right) mod2, 0 \leq j \leq 1 \qquad (10)$$

The previous summation represents a direct relationship between the input binary digits, and the CVNS digits. To obtain some insight on the previous expression, we expand this summation for two CVNS digits:

When i=0, $((x))_{j+t}$ becomes

$$((x))_0 = \left(\sum_{i=0}^{15} x_i . 2^i\right) mod2$$

$$= (x_0 + 2x_1 + 4x_2 + \cdots) mod2 = x_0 \qquad (11)$$

And for i=1

$$\begin{aligned}((x))_1 &= \left(\sum_{i=0}^{15} x_i . 2^{i-1}\right) mod2 \\ &= (2^{-1}x_0 + x_1 + 2x_2 + \cdots) mod2 \\ &= 2^{-1}x_0 + x_1 \qquad (12)\end{aligned}$$

Therefore, the general expression given by expression (10) can be modified to a pre congruent form to eliminate the modular reduction operator (mod2) as follows:

$$((x))_{j+t} = \sum_{i=0}^{j} x_{j+t} . 2^{i-j} \quad 0 \le j \le 15 \qquad (13)$$

From the system design point of view, this alteration not only simplifies the design and reduces the complexity, but it also decreases the delay of the adder. In this form, every CVNS digit is obtained directly from the input binary digits. In general, since the delay of D/A conversion is constant, and typically is less than the delay of a modular reduction circuit, the designed system tends to be faster. Using the previous expression, the digit wise summation of two CVNS numbers ((x)), and ((y)) is

$$\begin{aligned}((Z))_{j+t} &= \left(((x))_{j+t} + ((y))_{j+t}\right) \\ &= \left(\textstyle\sum_{i=0}^{j}(x_{i+t} + y_{i+t})\, 2^{i-j}\right) mod2 \qquad (14)\end{aligned}$$

The prior addition should have included the modular reduction unit, so the combined value of two CVNS digits may go outside the range of possible radix values. The block-level depiction of the improvements achieved in the CVNS adder is shown in Figures 2 and 3. Figure 2 depicts the initial state of the CVNS addition among two binary values when numbers are produced using, while Figure 3 illustrates the enhancements achieved by doing away with the continuous modular reduction operator..

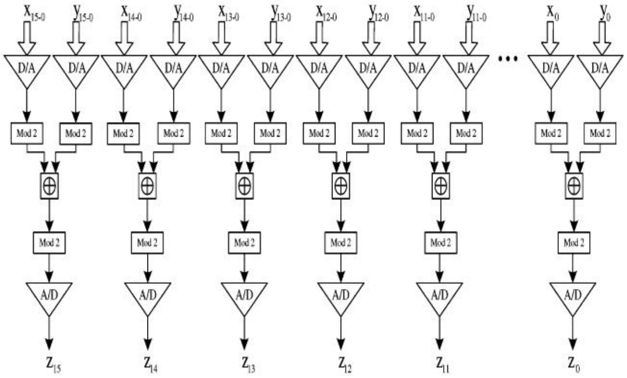


Fig. 2 Original Form of CVNS Addition

In this form, addition between two CVNS values requires conversion from binary to the CVNS representation, addition between the CVNS values, applying the modular reduction on the CVNS summation to adjust the values, and conversion back from the CVNS to binary representation. We are going to eliminate the modular reduction operation from the addition by manipulating the mathematical expressions to speed up the

ddition.

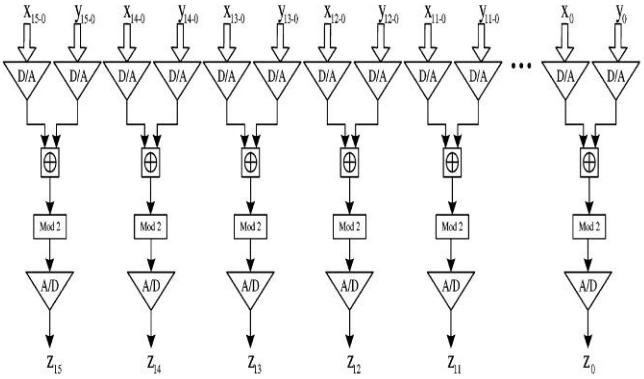


Fig. 3 Improvements made by manipulating the mathematical analysis of the CVNS addition, eliminating one gate in the addition

A CVNS digit as shown by the (10) is composed of two parts; an integer part and a non-integer part, which is shared with its less informed digits. By applying the (10) in the previous expression, the previous summation term can be expressed as follows:

$$((z))_{j+t} = \left( x_{j+t} + y_{j+t} + 2^{-j} \sum_{i=0}^{j-1} (x_{i+t} + y_{i+t}) B^{i-j} \right) mod2 \quad (15)$$

Or $$((z))_{j+t} = \left( x_{j+t} + y_{j+t} + 2^{-1} ((z))_{j+t-1} \right) mod2 \quad (16)$$

The term ((z)) $_{j+t}$ are in the CVNS form and have to be converted back to binary form. For values of ((z)) $_{j+t}$ € [0,1) binary outcome digit, ((z)) $_{j+t}$, is equal to 0, and for ((z)) $_{j+t}$ €[1,2) it is equal to 1. At this stage, low radix of the CVNS allows us to remove modular reduction operation (mod2) and CVNS to binary conversion with a simple XOR, and to generate the binary outcome of the adder as follows:

$$z_{j+t} = x_{j+t} \oplus y_{j+t} \oplus xy_{j+t} \quad (17)$$

Where $\oplus$ denotes the logical XOR function, and

$$xy_{j+t} = \begin{cases} 0, & if\ 0 \le 2^{-1}((z))_{j+t-1} < 1 \\ 1, & if\ 0 \le 2^{-1}((z))_{j+t-1} < 2 \end{cases} \quad (18)$$

It follows that we have simplified the system architecture and reduced the complexity of the adder by modifying the mathematical expressions of the CVNS addition. This implementation uses a mixed-signal format to add two binary values. Digital circuits calculate the end result, whereas analog values offer local carry information, minimizing the number of interconnections. In Figure 4, we can see the mixed-signal CVNS adder shown at the block level.

A more straightforward XOR gate stands in for the modular reduction and A/D conversion in the CVNS addition's output mixed signal. But before we can even think about building the adder circuitry, there's another major design problem that needs fixing. Since the CVNS representation is based on a continuous-value system, analog circuits are used to implement it. A high-resolution analog environment, in this case 14 bits, is necessary for the implementation of the CVNS adder. It has condition cannot be satisfied in most targeted analog technologies.

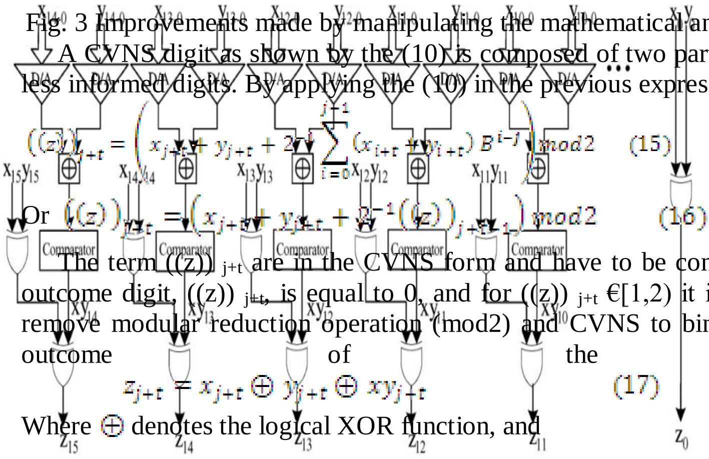


Fig. 4 Block level representation of the mixed-signal CVNS addition

If analog environment distinguishes $2^{\psi}$ different levels, in order to obtain a reliable value for the $((z))_{j+t+1}$,digit generation and addition has to be modified by performing the summation over a fixed-size group of bits of length ψ' with ψ >1.This parameter is technology dependent, limited by the maximum reliable resolution of environment. In this paper, we have chosen it to be equal to ψ=4, which not only can be easily implemented by reliable current-mode analog circuits in our target technology, but also simplifies the partitioning scheme. Addition over a group of digits called Truncated Addition [5], for 16-bit summation with is ψ=4 .

$$((z))_{j+t-1} = \left( \sum_{i=4\left\lceil \frac{j+2}{4} \right\rceil}^{j-1} (x_{i+t} + y_{i+t})\, 2^{i-j} + 2^{4\left\lceil \frac{j+2}{4} \right\rceil + 6 - j} Tr_{j+t+1} \right) mod\beta \ (19)$$

Where$Tr_{14-j+t}$is called the *truncation signal* and is equal to

$$Tr_{j+t+1} = \begin{pmatrix} gt_{(k+t)} + rt_{(k+t)} gt_{(k+t-1)} + .. \\ + rt_{(k+t)} rt_{(k+t-1)} .. rt_{(2+t)} rt_{(1+t)} C_{in_t} \end{pmatrix} \qquad (20)$$

Where *gt* and *rt* signals are

$$gt_{(k+t)} = \begin{cases} 1, & \text{if } \sum_{i=4k-4}^{4k-1} (x_{i+t} + y_{i+t}) 2^{i-4k+4} \geq 2 \\ 0, & \text{otherwise} \end{cases} \qquad (21)$$

$$rt_{(k+t)} = \begin{cases} 1, \text{if} & \sum_{i=4k-4}^{4k-1} (x_{i+t} + y_{i+t}) 2^{i-4k+4} \geq 1.875 \\ 0, & \text{otherwise} \end{cases} \ (22)$$

And *k* is an integer equal to $\left\lceil \frac{(11-j)}{4} \right\rceil$ and, *t* = 0, 16, 32, 48. In this approach, the required resolution of the analog environment is reduced to 4 bits; however, the arithmetic unit is able to process data with much longer word length. In fact, this form of the CVNS addition is similar to the carry-look-ahead concept in digital circuits, except that the addition is performed in analog domain. The truncation signals provide an estimate of the analog digit.

The resulted 64-bit adder is a mix of both classical binary circuits such as XOR for generating the output and CVNS style circuits for evaluating terms such as (24) and (25). Analog circuits in this design are the front circuits, and are used for processing a group of input bits with higher speed and fewer interconnections. These analog blocks detect the existence of the truncation signals within a group of binary inputs. Digital circuits are at the output stage of each adder and provide the required driving capability for various interconnection loadings in different adder Configurations.

Equation (23) indicates that only a limited number of truncation signals are required within the 16-bit adders. Because of the low number of interconnections, control and partitioning of the adders is performed with less complexity. The number of truncation signals depends on the chosen length of the groups. Based on the chosen group length in this design (*ψ=4),* in each of the 16-bit adders, the three truncation signals are as follows:

$$Tr_{t+4:t+7} = gt_{(1+t)} + rt_{(1+t)} C_{in_t} \qquad (23)$$

$$Tr_{t+8:t+11} =$$

$$ctrl8.in_{8+t} + ctrl8' \begin{pmatrix} gt_{(2+t)} + rt_{(2+t)} gt_{(1+t)} \\ + rt_{2+t} rt_{(1+t)} C_{in_t} \end{pmatrix}$$

(24)

$$Tr_{t+12:t+15} =$$

$$gt_{(3+t)} + rt_{(3+t)} \times \begin{pmatrix} ctrl8.in_{8+t} + \\ ctrl8' \times \begin{pmatrix} gt_{(2+t)} + rt_{(2+t)} gt_{(1+t)} + \\ rt_{2+t} rt_{(1+t)} C_{in_t} \end{pmatrix} \end{pmatrix}$$

(25)

Where $C_{in}$ is the carry input to each of the adders. The expression for this signal is derived in the next section. The three truncation signals given by the above equations are the only signals that are passed from the second layer of the adder to the third. The carry out of the adder is generated in the same style.

### *B. Radix- 2 64- bit Addition*

The reconfigurable 64-bit adder is generated by cascading four 16-bit radix-2 CVNS adders. This CVNS adder has a uniform design, making it suitable for reconfigurable media processing and SoC applications. In this adder, information is generated locally for addition hence; the number of required interconnections is reduced. The only global information is the carries out of each 16-bit adder. Between the four adders, there are eight truncation signals, which are similar to the *gt* and *rt* signals inside each adder, as follows:

$$\begin{aligned} gt_{(t:t+15)} = (&gt_{(t+12:t+15)} + rt_{(t+12:t+15)} gt_{(t+8:t+11)} \\ &+ rt_{(t+12:t+15)} rt_{(t+8:t+11)} gt_{(t+4:t+7)} + \\ &rt_{(t+12:t+15)} rt_{(t+8:t+11)} \times rt_{(t+4:t+7)} gt_{t:t+3}) \quad (26) \end{aligned}$$

$$rt_{(t:t+15)} = rt_{(t+12:t+15)} rt_{(t+8:t+11)} \times rt_{(t+4:t+7)} rt_{(t:t+3)} \quad (27)$$

Where t =0, 16, 32, 48

The propagation of these signals is controlled by *Ctrl* 32 and *Ctrl* 64. Input carry into each of the 16-bit adders, which is shown as *Cin* $_t$, is as follows:

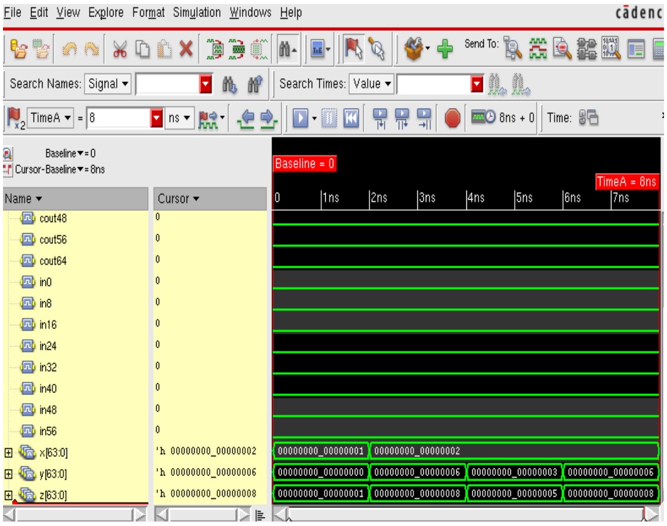
$$cin_{16} = crtl\,32' \left(gt_{(0.15)} + rt_{(0.15)} . in_0\right) + ctrl32 . in_{16} \qquad (29)$$

$$cin_{32} = crtl64' . in_{32} + crtl\ 64 \begin{pmatrix} gt_{(16:31)} + \\ rt_{(16:31)}\, gt_{(0:15)} \\ + rt_{(16:31)}\, rt_{(0:15)} . in_0 \end{pmatrix} (30)$$

$$cin_{48} = crtl32' . in_{48} + crtl\ 32 \begin{pmatrix} gt_{(32:47)} + \\ rt_{(32:47)} (crtl64' . in_{32} \\ + crtl\ 64 \begin{pmatrix} gt_{(16:31)} + \\ rt_{(16:31)} \\ gt_{(0:15)} \\ + rt_{(16:31)} \\ rt_{(0:15)} \\ . in_0 \end{pmatrix} \end{pmatrix} (31)$$

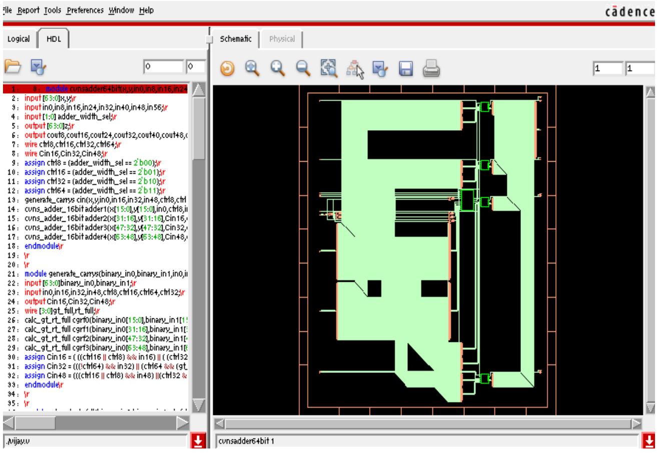
## V. RESULTS

Cadence analysis tools were used for simulating and synthesize RTL schematic diagram for the 64-Bit addition the proposed design Fig. 5 & Fig. 6. Comparing the ripple carry adder with CVNS adder, less power consumption is observed in CVNS adder Fig.7. The Table 3 shows Comparison between 64-bit Ripple Carry Adder and 64-bit CVNS Adder synthesis reports like area, timing and power.



```
cout48    0
cout56    0
cout64    0
in0       0
in8       0
in16      0
in24      0
in32      0
in40      0
in48      0
in56      0
x[63:0]   'h 00000000_00000002   00000000_00000001  00000000_00000002
y[63:0]   'h 00000000_00000006   00000000_00000000  00000000_00000006  00000000_00000003  00000000_00000006
z[63:0]   'h 00000000_00000008   00000000_00000001  00000000_00000008  00000000_00000005  00000000_00000008
```

Fig. 5 Simulated output of 64-bit mode



```
module cvnsadder64bit(x,y,in0,in8,in16,in24
input [63:0]x,y;\r
input in0,in8,in16,in24,in32,in40,in48,in56;\r
input [1:0] adder_width_sel;\r
output [63:0]z;\r
output cout8,cout16,cout24,cout32,cout40,cout48,c
wire ctrl8,ctrl16,ctrl32,ctrl64;\r
wire Cin16,Cin32,Cin48;\r
assign ctrl8 = (adder_width_sel == 2'b00);\r
assign ctrl16 = (adder_width_sel == 2'b01);\r
assign ctrl32 = (adder_width_sel == 2'b10);\r
assign ctrl64 = (adder_width_sel == 2'b11);\r
generate_carrys cin(x,y,in0,in16,in32,in48,ctrl8,ctrl
cvns_adder_16bit adder1(x[15:0],y[15:0],in0,ctrl8,i
cvns_adder_16bit adder2(x[31:16],y[31:16],Cin16,
cvns_adder_16bit adder3(x[47:32],y[47:32],Cin32,
cvns_adder_16bit adder4(x[63:48],y[63:48],Cin48,
endmodule\r
\r
\r
module generate_carrys(binary_in0,binary_in1,in0,i
input [63:0]binary_in0,binary_in1;\r
input in0,in16,in32,in48,ctrl8,ctrl16,ctrl64,ctrl32;\r
output Cin16,Cin32,Cin48;\r
wire [3:0]gt_full,rt_full;\r
calc_gt_rt_full cgrf0(binary_in0[15:0],binary_in1[1
calc_gt_rt_full cgrf1(binary_in0[31:16],binary_in1[
calc_gt_rt_full cgrf2(binary_in0[47:32],binary_in1[
calc_gt_rt_full cgrf3(binary_in0[63:48],binary_in1[
assign Cin16 = ( ((ctrl16 || ctrl8) && in16) || ( (ctrl32
assign Cin32 = (((!ctrl64) && in32) || (ctrl64 && (gt_
assign Cin48 = (((ctrl16 || ctrl8) && in48) ||(ctrl32 &
endmodule\r
\r
\r
```

Fig. 6 schematic diagram of 64-bit addition

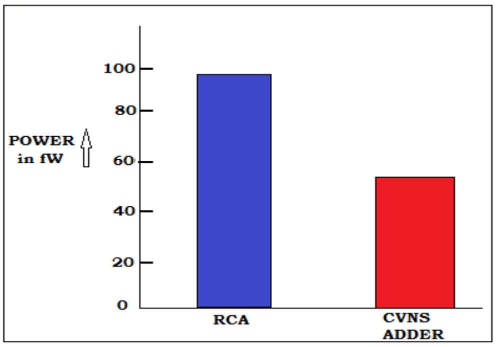


Fig. 7 Comparison of Power Consumption

TABLE 3
Comparison between 64-bit Ripple Carry Adder and 64-bit CVNS Adder synthesis reports

| Synthesis report | Area($\mu m^2$) | Power(fW) |
|---|---|---|
| RCA | 4558 | 98.55 |
| CVNS | 3995 | 59.87 |

## VI. Conclusion

The Verilog language has been used to build the CVNS adder. The 64-bit CVNS adder and the ripple carry adder were used to study the area and power consumption of the 64-bit adder, and the findings were compared. Due to its reduced power consumption, the suggested adder has been deemed superior than the ripple carry adder. Along with the aforementioned parameter improvement, future work may think about area reduction strategies employing 45 nm or smaller technology..

## References

[1] “Multiplexer based adder for media signal processing” by A. Farooqui, V. Oklobdzija, and F. Chechrazi was published in 1999 in the Proceedings of the IEEE International Symposium on Very Large Scale Integration Technology, Systems, and Applications, pages 100–103.

[2] [2] "A high speed energy efficient 64-bit reconfigurable binary adder," published in the IEEE Trans Very Large Scale Integer (VLSI) Syst journal in May 2003 by S. Perri, P. Corsonella, and G. Cocorullo.

[3] “A number system with continuous valued digits and modulo arithmetic” by A. Saed, M. Ahmadi, and G. A. Jullien was published in the November 2002 issue of the IEEE Transactions on Computers (vol. 51, no. 11, pp. 1294-1304).

[4]
[5] Digital multiplication utilizing continuous valued digits, by M. Mirhassani, M. Ahmadi, and G. A. Jullien, published in 2007 in Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), pages 3263–3266.
[6] In the 2005 IEEE International Symposium on Circuits and Systems (ISCAS), J.-F. Li, J.-D. Yu, and Y.-J. Huang presented an approach for designing reconfigurable hybrid carry-lookahead/carry select adders (vol. 1, pp. 77–80).
[7] In the 2006 Proc IEEE Asilomar Conf Sig, Syst, Comput, volume 9, pages 332–336, the authors Mirhassani, Ahmadi, and Jullien discuss 16-bit binary multiplication using high radix analog digits.
[8]