# Realization of Various Topologies of Adders Based on H-Dycml

**Riya Jain, Neeta Pandey**

***Abstract*****:** ***Power consumption minimization in a circuit becomes imperative with growth in demands of portable goods. However, at the same time, its speed limits the performance of a system. Therefore, there is a need of choosing optimum circuit architecture that takes into account the both conflicting parameters, that is, power dissipation and speed. Arithmetic unit is one of the vital components of portable goods and out of all arithmetic operations, adders are the most commonly used. To address the issue of high power dissipation, low-power designing styles are becoming prominent now-a-days. Hybrid Dynamic Current Mode Logic is high-speed, low-power designing style that has been recently proposed in literature. Therefore, this paper presents the comparison between performances of various topologies of adders that are implemented using a high-speed, low-power designing style: Hybrid-Dynamic Current Mode Logic (H-DyCML). All the circuits are realized in Cadence Virtuoso using 180nm CMOS technology parameter. Various performance parameters are evaluated such as: Delay, Power, Power-Delay Product, and hardware utilization. It is found that carry look-ahead adder out-stands other adders in terms of overall performance.***

***Keywords*****:** ***Ripple Carry Adder, Carry Skip Adder, Carry Look-Ahead Adder, Carry Save Adder, Carry Select Adder, H-DyCML***

## I. INTRODUCTION

Arithmetic circuits [1] find their use in wide range of applications and therefore, with growing number of applications, optimizing their performance becomes an important task to accomplish. For example, mostly all arithmetic operations such as multiplication, addition, and trigonometric function evaluation form a vital part in all the DSP (Digital Signal Processing) algorithms [2-3]. Since, DSP algorithms are the major components of telecommunication, and multimedia applications, they are quite important in embedded systems. Therefore, it is necessary to optimize power consumption and speed, especially for hand-held and portable devices. Out of all the arithmetic operations, addition is the most frequently used. Thus, this paper primarily focuses on adders and its different architectures. In order to reduce power consumption and obtain increased performance, researchers are getting inclined towards low power designs [4-8].

**Revised Manuscript Received on April 30, 2020.**
***** Correspondence Author

**Riya Jain***, Electronics and Communication Department, Delhi Technological University, Delhi, India. Email: riyajain_bt2k16@dtu.ac.in
**Neeta Pandey**, Electronics and Communication Department, Delhi Technological University, Delhi, India. Email: n66pandey@rediffmail.com

Generally, conventional CMOS logic is used in digital integrated circuits. However it poses a limitation due to high dynamic power dissipation. This limited the use of conventional CMOS logic and made researchers get inclined towards MOS current mode logic (MCML)-based [9-11] circuits. This type of logic uses differential input and gives differential output with reduced voltage swing. Due to reduced voltage swing, the dynamic power dissipation in MCML was reduced significantly. However, it had static power dissipation due to presence of static current source. To overcome the limitation of static power dissipation, further modifications were made and dynamic current mode logic (DyCML) circuits [12-13] were introduced. In DyCML circuits, constant current source was changed into dynamic current source due to which static power dissipation was minimized. However, certain limitations on delay still persist due to stacking of transistors. This led to introduction of Hybrid-Dynamic Current Mode Logic (H-DyCML) circuits [14] which make use of complementary pass transistor logic (CPL) [15-16] to realize a logic function. It eliminates the need of multiple-layered source coupled transistors and thus reduces the delay. In this paper, H-DyCML has been used to implement all the topologies of adders and thus compare their performance. H-DyCML has been explained in section 2 and various implemented topologies of adders are explained in section 3. Section 4 presents the results of and comparison between various architectures of adders. Finally, the paper is concluded in section 5, followed by the references.

## II. HYBRID DYNAMIC CURRENT MODE LOGIC (H-DYCML) CIRCUITS

First step taken to overcome the dynamic power dissipation in conventional CMOS circuits was the introduction of MOS Current Mode Logic (MCML). However, in an MCML circuit, a constant current source was used which posed another limitation of static power dissipation. To address this issue, Dynamic Current Mode Logic (DyCML) circuits [13] were proposed which made use of dynamic current source replacing the static current source in MCML circuits. Apart from modifying the current source, load resistors were also replaced with PMOS transistors connected with cross coupled PMOS latches. A pull down network is used to realize the logic functions which consists of source coupled transistors stacked together, driven by differential inputs. This stacking of transistors leads to increase in delay of the implemented logic which is why [14] proposes a new method that resolves the issue of stacking. This method is termed as Hybrid Dynamic Current Mode Logic (H-DyCML) and it realizes the function using complementary pass transistor logic (CPL) [15].

Basic architecture of H-DyCML circuits is shown in Fig. 1. The operation of H-DyCML circuits is similar to that of DyCML circuits and it is

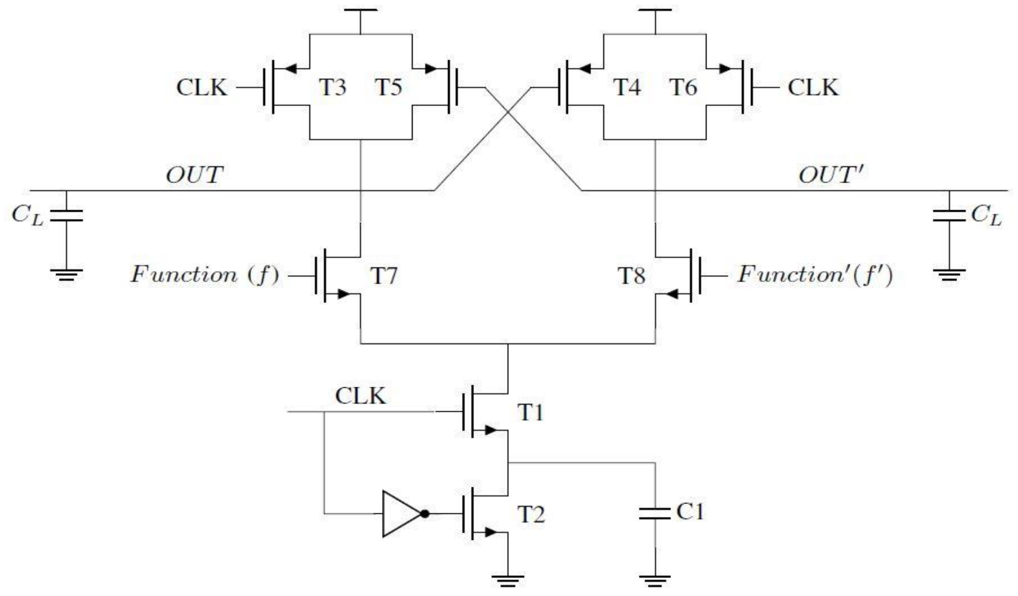

**Fig. 1 : Basic Architecture of H-DyCML circuits [14]**

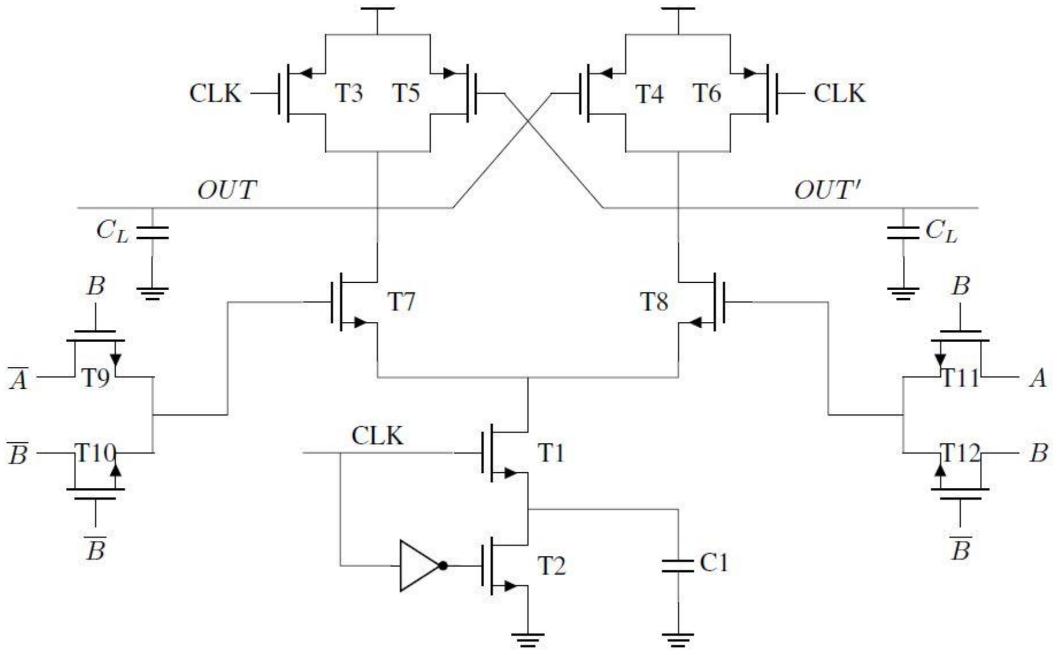

**Fig. 2 : Realization of AND gate using H-DyCML style [14]**

explained further. H-DyCML circuits follow the mechanism of pre-charge and evaluation phases. This is a principle defined for dynamic CMOS circuits [15,16]. In pre-charge phase, the output nodes, OUT and OUT', are charged to a voltage level of $V_{DD}$ in advance. This is possible because in pre-charge phase CLK = LOW which is why transistors T3 and T6 are turned ON. This allows a direct path between output nodes and supply voltage ($V_{DD}$). At the same time, transistor T2 is also turned ON while turning OFF T1. Thus, capacitor C1 discharges to the ground potential through T2. Since T1 act as an open switch, there is no path between output nodes and ground via transistors T7 and T8. Therefore, any input applied to these transistors does not affect the potential of output nodes. It can be seen in Fig. 1 that inputs of T7 and T8 are driven by a function (f) and its complement function (f') respectively. In H-DyCML circuits, these functions are realized using a complementary pass transistor logic (CPL) network. Thus, outputs of these CPL network do not affect the output nodes during pre-charge phase. They are capable of affecting the output nodes in evaluation phase when a direct path is created between the output nodes and the ground via transistors T7 and T8. This is possible because in evaluation phase CLK becomes HIGH which results in switching OFF of transistors T3, T6 and T2. At the same time, transistor T1 is turned ON which facilitates direct connection of output nodes and the capacitor C1 which discharged to ground potential in pre-charge phase.

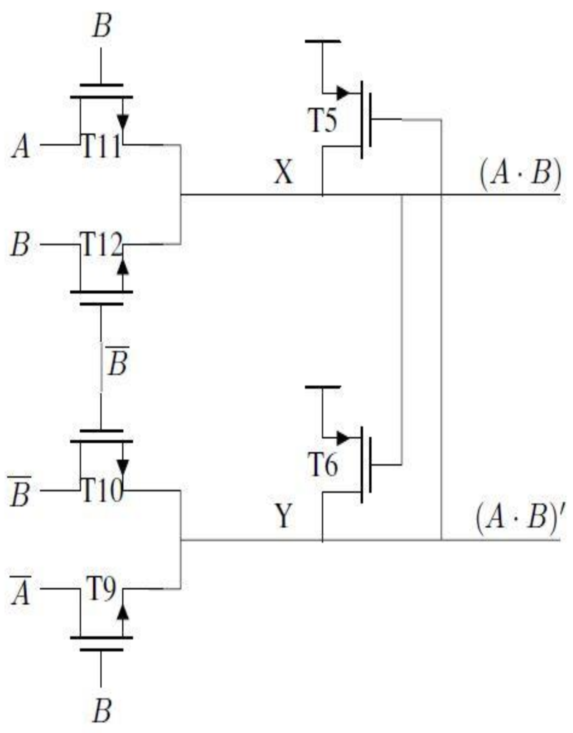

**Fig. 3 : Technique of Level Restoring [14,15]**

Thus, during evaluation, any input applied to the CPL network that affects its output also affects the output nodes of H-DyCML circuit. At this time, total charge in load capacitors ($C_L$) is redistributed among two load capacitors and C1. Since the two transistors, T7 and T8, are driven by two complementary functions, one of them is turned ON and the other one is turned OFF due to which one of the output node remain charged to the potential $V_{DD}$ and the other output node discharges to a potential of $V_{DD} – V_{SWING}$. Simultaneously, the capacitor C1 charges from ground potential to a voltage level of $V_{DD} – V_{SWING}$. One thing to be mentioned here is that in H-DyCML circuits, the value of C1 depends on the value of load capacitors ($C_L$) and it is evaluated by the equation given as:

$$C1 = \frac{V_{swing}}{V_{DD} - V_{swing}} C_L \quad (1)$$

An example is shown in Fig. 2 in which 2-input AND gate is realized using H-DyCML style [14]. In the shown figure, complementary pass transistor logic is used to implement AND and NAND logic as explained earlier. Transistors T11 and T12 realizes the AND (A · B) logic such that whenever the output of this network is HIGH, OUT’ is pulled down to ground potential. Similarly, T9 and T10 transistors realize NAND (A · B’) logic and whenever output of AND logic is HIGH, NAND logic will be LOW. Thus, transistor T7 will be turned OFF and OUT stays at voltage level $V_{DD}$. However, it is a well-known fact that NMOS transistors are capable of offering weak pull-up and strong pull-down. Due to this property, maximum possible voltage at the output of NMOS pass transistor is $V_{OH} = V_{DD} – V_{T,n}$, that is, gate voltage ($V_{DD}$) is reduced by one threshold voltage. Therefore, this can lead to erroneous results. To avoid such situation, 3 different methods have been proposed in [14]. Out of these three methods, we used the technique of level restorer which can be explained using Fig. 3. In the technique of level restorer, 2 cross coupled PMOS transistors are used. Since PMOS pass transistors have the property of offering strong pull up, they are designed such that they pulls up the voltage level of output of CPL network to $V_{DD}$ whenever the output of complementary logic is LOW. However, there are certain limitations with this method, that is, the voltage swing ($V_{SWING}$) should be greater than the threshold voltage of the PMOS transistor ($V_{T,p}$) in order to turn ON the transistors. Yet, we chose this method over other two methods because it is easier to handle the voltage swing without manipulating the threshold voltages or increasing the number of transistors.

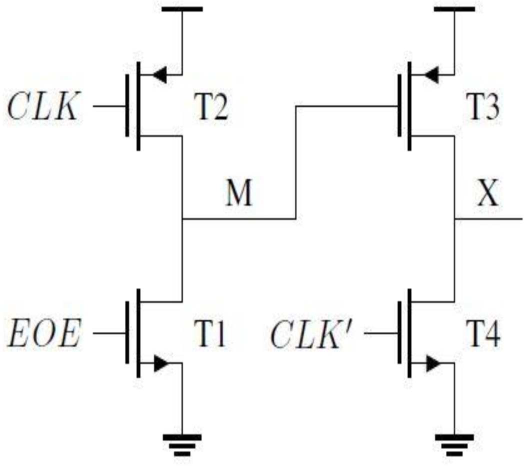

**Fig. 4 : Self-Timed Buffer [13]**

Realization of individual gates is relatively simpler than realization of two or more cascaded gates such that output of first gate is applied to input of second gate. The problem that is faced in cascading is that if we directly connect the output of one gate to the input of another, then both the gates have their pre-charge and evaluation phase occurring simultaneously. This leads to erroneous results because second stage starts evaluating before the first stage has completed its evaluation. Therefore, to address this issue, a self-timed buffer has been proposed in [13] which is an additional circuitry that is added between the two stages. It adds some delay before the second stage and after the first stage. The circuit has been shown in Fig. 4 where EOE is the voltage across capacitor C1 of previous stage. Its working has been explained further. In pre-charge phase of H-DyCML circuits, when CLK = LOW, capacitor C1 is discharged to ground potential due to which EOE = LOW. Thus, transistor T1 is turned OFF and T2 is turned ON. Therefore, node M is pull up to voltage level $V_{DD}$ and subsequently, transistor T3 is turned OFF. At the same time, T4 is turned ON because CLK’ is HIGH which results in lowering of potential of node X to ground. During evaluation phase, CLK = HIGH which is why T2 is turned OFF and since, capacitor C1 of H-DyCML circuit is charged up to a voltage level of $V_{DD} – V_{SWING}$ , EOE becomes HIGH. It results in switching ON of transistor T1 and subsequently, node M is pulled down to ground potential which further results in switching ON of T3. Simultaneously, T4 is turned OFF and node X is pulled up to a voltage $V_{DD}$ through transistor T3. Therefore, CLK signal gets transmitted to node X with some added delay.

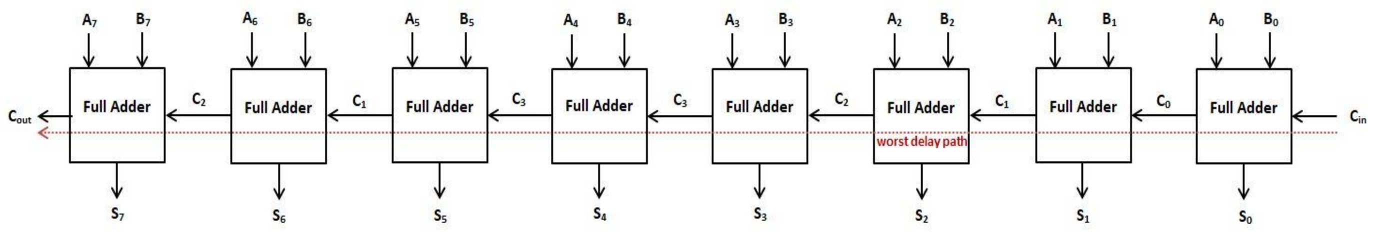

**Fig. 5: Ripple Carry Adder [1,16]**

## III. TOPOLOGIES OF ADDERS

To implement all the adders, above explained designing style has been considered. Various architectures of adders realized using H-DyCML style has been explained in this section.

### A. Ripple Carry Adder

Ripple carry adder is the most basic adder whose architecture is shown in Fig. 5 which presents an 8-bit ripple carry adder. It consists of 8 full adders arranged in series such that carry output from previous stage is fed as carry input in next stage. In this type of adder, the delay increases with increase in number of bits. Worst delay path has been shown in the Fig. 6 with the dotted line and it can be seen that worst case delay is when carry ripples through every stage. It is possible when A = 11111111, B = 00000001 and $C_{in}$ = 0 for an 8-bit adder.

### B. Carry Look-Ahead Adder

Carry look-ahead adder provides a solution for increased delay in case of a ripple carry adder due to propagation of carry through every stage. This issue is resolved by evaluating carry signals in advance and in parallel based on the input signals. The value does not depend upon the value of output carry of previous stage. The architecture for 8-bit carry look-ahead adder is shown in Fig. 6 where the carry generate block generates the carry signals. The carry signals generated can be understood with the help of the equations explained further. In a half adder:

$$P_i = A_i \oplus B_i \quad (2)$$

$$G_i = A_i \cdot B_i \quad (3)$$

And, for a full adder,

$$S_i = P_i \oplus C_i \quad (4)$$

$$C_{i+1} = G_i + P_i \cdot C_i \quad (5)$$

Therefore, if we extend these equations for a 4 bit adder, we get values of $C_1$, $C_2$, $C_3$, and $C_4$ as follows:

$$C_1 = G_0 + P_0 C_0 \quad (6)$$

$$C_2 = G_1 + P_1 G_0 + P_1 P_0 C_0 \quad (7)$$

$$C_3 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0 \quad (8)$$

$$C_4 = G_4 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_0 \quad (9)$$

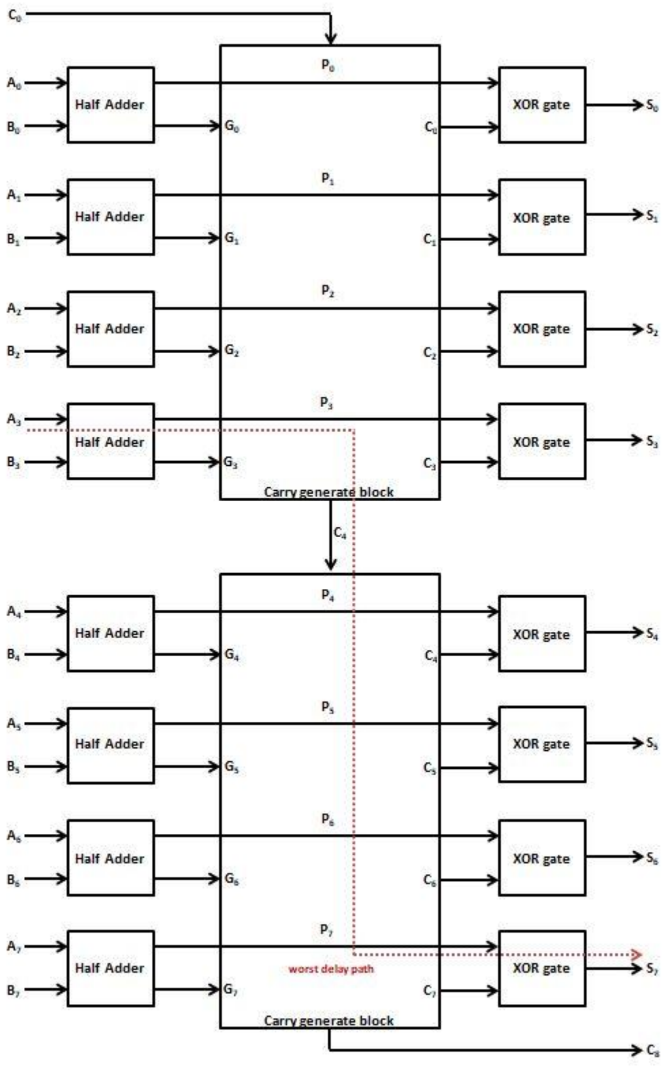
C0
A0 B0 A1 B1 A2 B2 A3 B3 A4 B4 A5 B5 A6 B6 A7 B7
Half Adder
P0 G0 C0 P1 G1 C1 P2 G2 C2 P3 G3 C3
Carry generate block
C4
P4 G4 C4 P5 G5 C5 P6 G6 C6 P7 G7 C7
worst delay path
Carry generate block
XOR gate
S0 S1 S2 S3 S4 S5 S6 S7
C8

**Fig. 6: Carry Look-Ahead Adder [1,16]**

Therefore values of $C_2$, $C_3$ and $C_4$ do not depends on the value of carry generated by previous stage, but only the value of input carry. To implement the 8-bit adder, 4-bit carry look-ahead adder is connected in ripple carry fashion. The worst delay path for carry look-ahead adder has also been shown in Fig. 6 with a dotted line and it can be seen that the delay is equal to sum of delay of one half adder, one xor gate and that of two carry generate blocks.

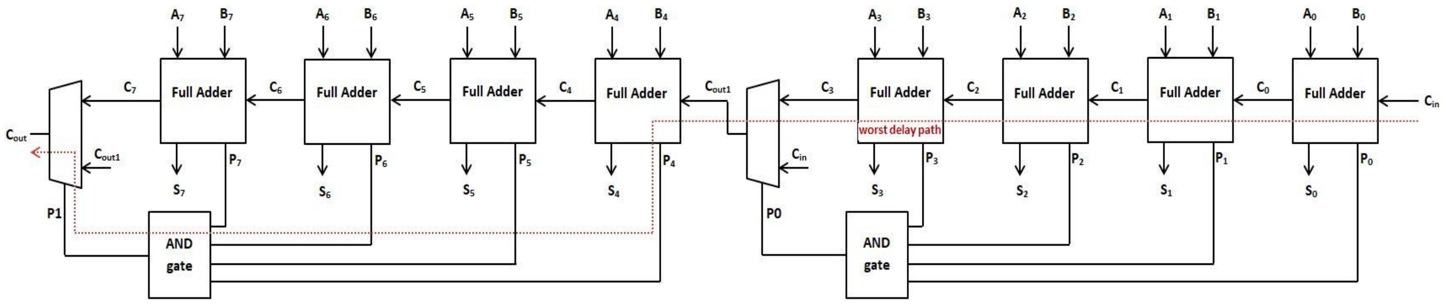

**Fig. 7: Carry Skip Adder [1,16]**

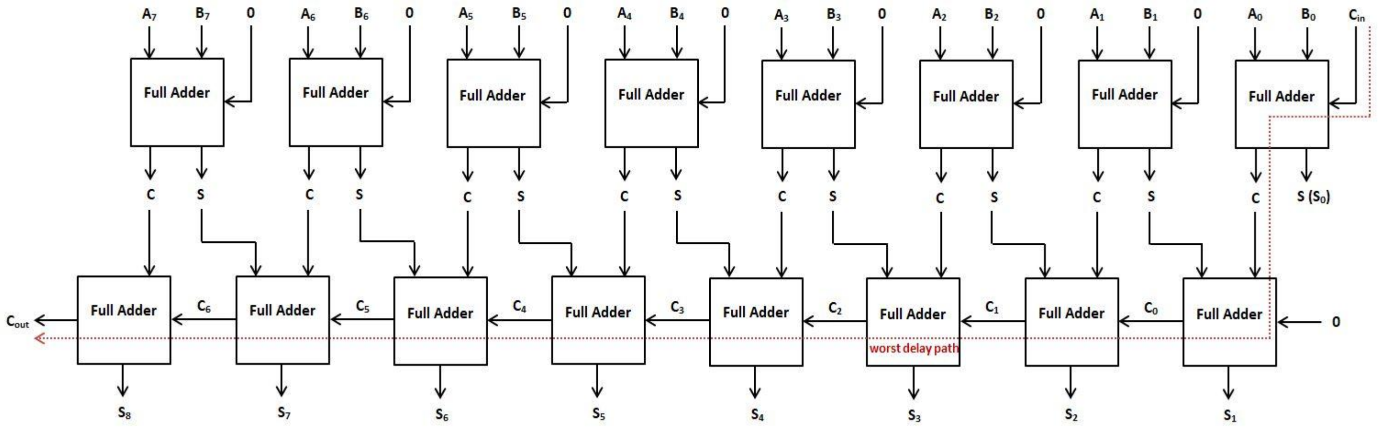

**Fig. 8: Carry Save Adder [1,16]**

## A. Carry Skip Adder

Carry skip adder is similar to ripple carry adder with a slight modification in how output carry signal is generated. It does not wait for the carry bit to ripple through the entire chain of full adders, rather uses a skip chain to speed up the generation of carry signal. Fig. 7 shows the architecture of 8-bit carry skip adder. 4-input AND gate and a 2x1 Multiplexer forms the skip chain and the logic can be explained by following equations. The select line (P0 or P1) that selects the correct value of carry is determined by:

$$P0 = P_3 \cdot P_2 \cdot P_1 \cdot P_0 \qquad (10)$$

And,

$$P1 = P_7 \cdot P_6 \cdot P_5 \cdot P_4 \qquad (11)$$

Therefore, if P0 = 0, then the value or carry is determined by the carry output of the first ripple carry adder ($C_3$). Otherwise, value of carry is same as the input carry bit ($C_{in}$). Similarly, if P1 = 0, then $C_7$ is chosen, otherwise $C_{out1}$. For example, if A = 11111111, B = 00000001, and $C_{in}$ = 0, then select line (P0) is equal to 0 and P1 is equal to 1. Thus, value of $C_3$ determines the value of ($C_{out1}$) and it will be propagated to $C_{out}$ by skipping the ripple carry adder. The worst case delay path has been shown in Fig. 7 with a dotted line and it is equal to the delay of one ripple carry adder, one 4-bit AND gate, and two multiplexers. The delay of 8-bit carry skip adder is less than delay of an 8-bit ripple carry adder because the carry bit skips the ripple carry adder and is directly generated using skip chain. However, the delay of carry skip adder is greater than ripple carry adder for 4-bit adders [16].

## B. Carry Save Adder

Carry save adder has architecture as shown in Fig 8. As seen in the figure, carry save adder adds 3 bits at a time using a full adder without propagating the carry output. It reduces n-bit m-operands addition to n-bit 2-operand addition by saving the carry and sum bits. It then uses ripple carry adder to add sum and carry bits in last stage. Fig. 8 shows the 8-bit ripple carry adder where bits of A, B and $C_{in}$ are added in parallel and their sum and carry bits are saved. In next stage, the carry bits are shifted left by one place and then added with sum bits.

The worst delay path has been shown in the figure with a dotted line and it can be seen that delay is equal to delay of one full adder and one 8-bit ripple carry adder. The carry save adders are useful for additions of more than two operands where it reduces the delay significantly. However, in this paper, two operand additions is carried forward which is why the delay of carry save adder is greater than ripple carry adder. One thing to be noted here is that the number of bits in output of carry save adder increases with increase in carry save stages. For example, 10-bit output is obtained for 8-bit carry save adder as shown in Fig. 8.

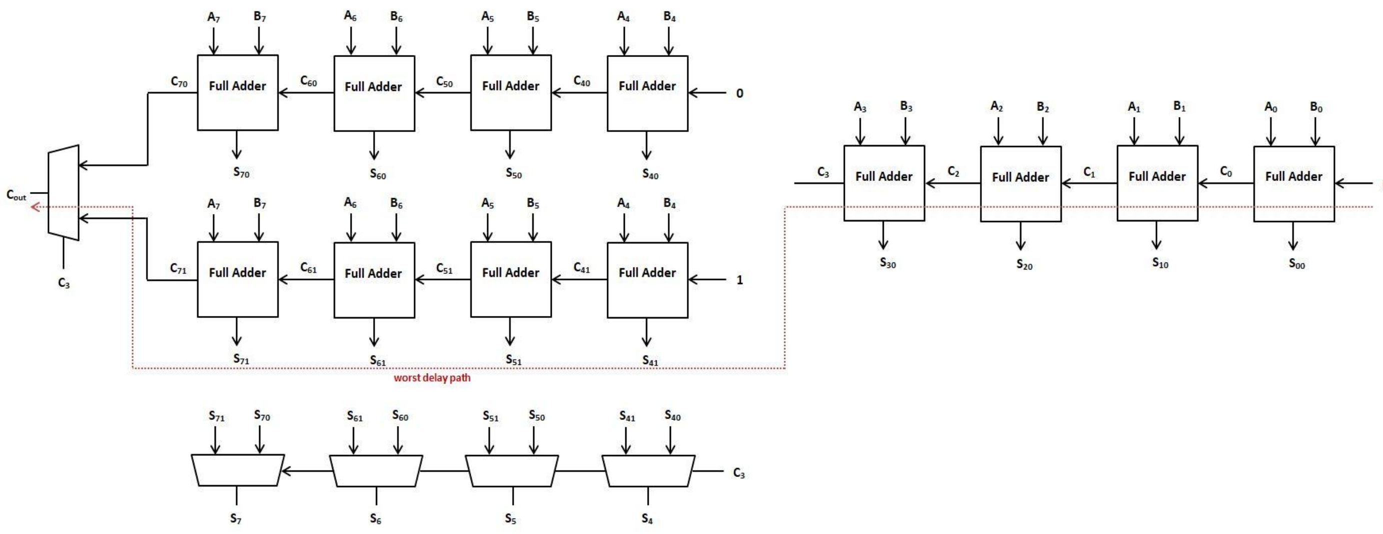

**Fig. 9: Carry Select Adder [1,16]**

**Table- I: Performance parameters for 8-bit adders designed using H-DyCML**

| **Parameters** | **Style** | | | | |
|---|---|---|---|---|---|
| | ***Ripple Carry Adder*** | ***Carry Skip Adder*** | ***Carry Look-Ahead Adder*** | ***Carry Save Adder*** | ***Carry Select Adder*** |
| Power (uW) | 213.840 | 267.084 | 230.914 | 399.732 | 333.992 |
| Delay (ns) | 9.532 | 6.981 | 4.869 | 10.872 | 5.406 |
| PDP (pJ) | 2038.276 | 1864.486 | 1124.390 | 4345.838 | 1805.483 |
| STB Count | 23 | 27 | 12 | 40 | 34 |
| Gate Count | 40 | 52 | 52 | 80 | 65 |
| Transistor Count | 778 | 1002 | 1024 | 1520 | 1244 |

### C. Carry Select Adder

In carry select adder, 2 ripple carry adders are used in parallel such that one of them takes 0 as the input carry bit and the other takes 1. The output bits of both the ripple carry adders are fed as input to 2x1 multiplexers connected at the end which uses true value of input carry as the select line. Therefore, additions are carried out twice and multiplexers are used for selecting the right values of sum and carry bits. The architecture for the same is shown in Fig. 9 and the worst delay path can be seen in the figure depicted with a dotted line. As shown in Fig. 9 in carry select adder, the delay is reduced as compared to ripple carry adder because it equal to sum of the delay of one 4-bit ripple carry adder and one multiplexer.

## IV. RESULTS AND COMPARISON

3 different gates are required in designing half and full adders which are building blocks of all topologies of adders. All the gates are designed using H-DyCML style using 180nm CMOS technology parameters. These circuits are simulated using Cadence Virtuoso software. As discussed in section 2, $V_{swing}$ must be greater than $V_{T,p}$ to avoid inconsistent results at the output of CPL network. Therefore, to satisfy this condition, $V_{swing}$ is chosen to be equal to 700mV. The supply voltage in all the circuits is taken to be 1.8V. Thus, value of $V_{DD}$ - $V_{swing}$ comes out to be 1.1V. Apart from this, load capacitor value ($C_L$) is taken to be 40fF and using (1), C1 assumes the value which is approximately equal to 26fF. Various adders implemented for comparison are: ripple carry adder, carry look-ahead adder, carry skip adder, carry save adder, and carry select adder. For all the circuits, the parameters mentioned above are kept the same.

For uniformity, 8-bit 2-operands operation was considered for all the adders such that A = 11111111, B = 00000001, and $C_{in}$ = 0. The valued of A, B, and $C_{in}$ are chosen so that carry is propagated through all the stages. Different parameters compared for all the adders are: Delay, Power and Power Delay Product (PDP). Delay for all the implemented architectures are calculated for the worst case, that is, when the carry is propagated through the chain of full adders. It is calculated as the time difference between when the inputs are applied and when we get the last output bit. For various adders, the path followed in the worst case is shown in their respective architectures. Average power [16] is calculated for all the adders to take into account the static and dynamic power. It is calculated using the following equation:

$$P_{avg} = \frac{1}{T}\int_0^T p(t)d(t) = \frac{V_{\sup ply}}{T}\int_0^T I_{\sup ply}(t)dt \qquad (12)$$

The results of various performance parameters such as delay, power, PDP, and hardware utilization for different adder topologies are recorded in Table 1 from where certain observations can be made:

- Power consumption is the least for ripple carry adder and maximum of carry select adder. The reason behind this is that the hardware utilization in carry save adder is almost double of that in ripple carry adder.

- Delay is least in case of carry look-ahead adder due to the fact that the intermediate carry bits are generated in parallel and are independent of output carry bit of previous stage. Delay is highest for carry save adder because we performed 2-operands addition for all the adders.
- PDP is the least for carry look-ahead adder and it stands out in terms of performance from all the other adders.
- Carry look-ahead adder uses the least number of self-timed buffers which accounts for its minimum delay.
- Delay of carry select adder is less than ripple carry adder even though it uses more number of self-timed buffers. This is because the computation in carry select adder occurs in parallel. Similarly, in case of carry skip adder, delay is less than ripple carry adder because it skips the propagation of carry bit through all stages by making use of skip chain.
- Delay for carry skip adder and carry select adder is less than that of ripple carry adder. But this is possible only on the cost of increased hardware (gate count).
- Ripple carry adder has least transistor count, that is, least area utilization which accounts for its least power consumption. Similarly, carry save adder has the highest transistor count which explains its highest power consumption.

## V. CONCLUSION

This paper presents the performance comparison for five different adders for 2-operand addition. Various adders compared are: ripple carry adder, carry skip adder, carry look-ahead adder, carry save adder, and carry select adder. All the gates used for realization of adders are designed using hybrid dynamic current mode logic (H-DyCML) style. H-DyCML circuits improve on the delay and power consumption as compared to DyCML. These circuits are designed in Cadence Virtuoso using 180nm CMOS technology parameters. Therefore, it will be right to conclude here, on the basis of data presented in previous section, that carry look-ahead adder provides overall better performance as compared to other topologies. It has the least delay and PDP. Moreover, it utilizes least number of STB and reasonably lower number of gate counts.

## REFERENCES

1. B. Parhami, “Computer Arithmetic: Algorithms and Hardware Designs”, 2nd edition, Oxford University Press, New York, 2010.
2. C. Pan, Y. Song, Z. Wang and C. Sechen, “DSP power reduction through Generalized Carry-Save arithmetic,” 2009 IEEE Dallas Circuits and Systems Workshop (DCAS), Richardson, TX, 2009, pp. 1-4.
3. S. Bhattacharjee, S. Sil, B. Basak and A. Chakrabarti, ``Evaluation of power efficient adder and multiplier circuits for FPGA based DSP applications," 2011 International Conference on Communication and Industrial Application, Kolkata, West Bengal, 2011, pp. 1-5.
4. B. Davari, R. H. Dennard and G. G. Shahidi, "CMOS scaling for high performance and low power-the next ten years," in Proceedings of the IEEE, vol. 83, no. 4, pp. 595-606, April 1995.
5. A. P. Chandrakasan, S. Sheng and R. W. Brodersen, "Low-power CMOS digital design," in IEEE Journal of Solid-State Circuits, vol. 27, no. 4, pp. 473-484, April 1992.
6. A.P. Chandrakasan, and R.W. Broderson, "Low Power Digital CMOS Design," Springer Science+Business Media New York, 1995.
7. R. H. Krambeck, C. M. Lee and H. -F. S. Law, "High-speed compact circuits with CMOS," in IEEE Journal of Solid-State Circuits, vol. 17, no. 3, pp. 614-619, June 1982.
8. P. Ng, P. T. Balsara and D. Steiss, "Performance of CMOS differential circuits," in IEEE Journal of Solid-State Circuits, vol. 31, no. 6, pp. 841-846, June 1996.
9. A. H. Ismail and M. I. Elmasry, "A low power design approach for MOS current mode logic," IEEE International [Systems-on-Chip] SOC Conference, 2003. Proceedings., Portland, OR, USA, 2003, pp. 143-146.
10. M. Alioto and G. Palumbo, "Feature - Power-aware design techniques for nanometer MOS current-mode logic gates: a design framework," in IEEE Circuits and Systems Magazine, vol. 6, no. 4, pp. 42-61, Fourth Quarter 2006.
11. M. Yamashina, and H. Yamada, "An MOS Current Mode Logic (MCML) Circuit for Low-Power Sub-GHz Processors," IEICE Transactions on Electronics, Vol. E75-C, pp. 1181–1187, 1992.
12. M. W. Allam and M. I. Elmasry, "Dynamic current mode logic (DyCML), a new low-power high-performance logic family," Proceedings of the IEEE 2000 Custom Integrated Circuits Conference (Cat. No.00CH37044), Orlando, FL, USA, 2000, pp. 421-424.
13. M. W. Allam and M. I. Elmasry, "Dynamic current mode logic (DyCML): a new low-power high-performance logic style," in IEEE Journal of Solid-State Circuits, vol. 36, no. 3, pp. 550-558, March 2001.
14. N. Pandey, D. Garg, K. Gupta, and B. Choudhary, "Hybrid Dynamic MCML Style: A High Speed Dynamic MCML Style," Journal of Engineering, Vol. 2016.
15. S-M Kang, and Y. Leblebici, "CMOS Digital Integrated Circuits: Analysis and Design," McGraw-Hill Higher Education, 3rd edition, 2003.
16. J. M. Rabaey, A. Chandrakasan, and B. Nikolic, "Digital Integrated Circuits: A Design Perspective," Pearson Education, Inc, 2nd edition, 2003.

## AUTHORS PROFILE

**Riya Jain** is currently pursuing her B. Tech in the discipline Electronics and Communication Engineering from Delhi Technological University (DTU). She is a research enthusiast who aims to contribute in the field of Electronics and Communication in coming future. Her areas of interest are: Digital VLSI design, Digital Design, and Computer Architecture. She has been contributing to the field of Electronics since 2016 and has been rigorously involved in research since 2018. Being a novice researcher, she is still exploring multiple domains in the field of Electronics and is motivated enough for further contribution to technological advancement in the same.

**Prof. Neeta Pandey,** M. E. (Microelectronics) Ph.D. She is a professor at Electronics and Communication Department, Delhi Technological University. She has a teaching and research experience of approximately 30 years. Being a research enthusiast in Electronics and Communication Engineering, she has more than 250 international journal publications. Her areas of interest are Analog and Digital VLSI Design, Computer Architecture and Current mode ADC Design. She is a member of multiple professional societies such ass IEEE, Women in Engineering (WIE), and affinity group of IEEE, ISTE. Furthermore, she is the mentor of Digital System Architecture and Design (DSAD) Research Group of Delhi Technological University.