

# SPST Using Multiple Domino Logic High Speed CLA Adder

Author: Hariharan.P; Tamilarasan.G

Affiliation: PG scolar ; Sree Sakthi Engineering college Coimbatore-641104

E-mail: harimehara@gmail.com

## ABSTRACT

The addition of two binary numbers is the fundamental and most often used arithmetic operation on microprocessors, digital signal processors (DSP), and data processing application-specific integrated circuits (ASIC). Therefore, binary adders are crucial building blocks in very large-scale integrated (VLSI) circuits. Their efficient implementation is not trivial because a costly carry propagation operation involving all operand bits has to be performed.

In this brief, an efficient implementation of an 8-bit Manchester carry chain (MCC) adder in multi output domino CMOS logic is proposed. The carries of this adder are computed in parallel by four independent 4-bit carry chains. Due to its limited carry chain length, the use of the proposed 8-bit adder module for the implementation of wider adders leads to significant operating speed improvement compared to the corresponding adders based on the standard 4-bit and MCC adder module.

The above proposed system I used in the application of a spurious-power suppression technique (SPST) which can dramatically reduce the power dissipation of combinational VLSI designs for multimedia/DSP purposes. The proposed SPST separates the target designs into two parts, i.e., the most significant part and least significant part (MSP and LSP), and turns off the MSP when it does not affect the computation results to save power.

**Keywords:** High speed adder, SPST, Carry look Ahead Adder, VLSI, Area Power Minimization.

## 1. INTRODUCTION

Addition is the most commonly used arithmetic operation and also the speed-limiting element to make faster VLSI processors. As the demand for higher performance processors grows, there is a continuing need to improve the performance of arithmetic units and to increase their

functionality. High-speed adder architectures include the carry look ahead (CLA) adders, carry-skip adders, carry-select adders, conditional sum adders, and combinations of these structures [1]–[4]. High-speed adders based on the CLA principle remain dominant, since the carry delay can be improved by calculating each stage in parallel. The CLA algorithm was first introduced in [5], and several variants have been developed. The Manchester carry chain (MCC) is the most common dynamic (domino) CLA adder architecture with a regular, fast, and simple structure adequate for implementation in VLSI [6], [7]. The recursive properties of the carries in MCC have enabled the development of multioutput domino gates, which have shown area-speed improvements with respect to single-output gates.

In this brief, a new 8-bit carry chain adder block in multioutput domino CMOS logic is proposed. The even and odd carries of this adder are computed in parallel by two independent 4-bit carry chains. Implementation of wider adders based on the use

## 2. PRELIMINARY CONCEPTS

Let  $A = a_{n-1}a_{n-2} \cdots a_1a_0$  and  $B = b_{n-1}b_{n-2} \cdots b_1b_0$  represent two binary numbers to be added and  $S = s_{n-1}s_{n-2} \cdots s_1s_0$  be their sum. In the following, the symbols  $\cdot$ ,  $+$ ,  $\oplus$  and  $-$  are used to denote the AND, INCLUSIVE OR, EXCLUSIVE OR, and NOT logical operations, respectively. In binary addition, the computation of the carry signals is based on the following recursive formula:

$$c_i = g_i + z_i \cdot c_{i-1} \quad (1)$$

where  $g_i = a_i \cdot b_i$  and  $z_i$  are the carry generate and

the carry propagate terms, respectively. The latter, for the case of INCLUSIVE OR adders, is defined as  $z_i = t_i = a_i + b_i$ , while for the case of EXCLUSIVE OR adders, it is defined as  $z_i = p_i = a_i \oplus b_i$ . In Fig. 1, the implementation of the generate and the two types of propagate signals in domino CMOS logic is shown.

Expanding relation (1), each carry bit  $c_i$  can be expressed as

$$c_i = g_i + z_i g_{i-1} + z_i z_{i-1} g_{i-2} + \dots + z_i z_{i-1} \dots z_1 g_0 + z_i z_{i-1} \dots z_0 c_{-1}. \quad (2)$$

The sum bits of the adder are defined as  $s_i = p_i \oplus c_{i-1}$ , where  $c_{-1}$  is the input carry.

The MCC [6], [7] generates all the carries computed according to relation (2) in parallel, using an iterative shared transistor structure. In practice, the CLA length is limited to four in order to cut down the number of series-connected transistors. Fig. 2 shows the conventional implementation of the 4-bit carry chain using multioutput domino CMOS logic.

MCC adders are adders, i.e., the carry propagate signal is defined as  $z_i = p_i = a_i \oplus b_i$ , to avoid false discharges produced at the output nodes of the carry chain due to higher OR-AND forms of multioutput gates.

For the implementation of the sum signals, the domino chain is terminated, and the sum bits of the MCC adder are the MCC is supported by the carry-skip capability to improve performance.

### 3. DOUBLE CARRY CHAIN ADDERS

MCC adders can efficiently be designed in CMOS logic. As mentioned previously, due to technological constraints, the length of their carry chains is limited to 4 bits. However, these 4-bit adder blocks are used extensively in the literature [2], [7], [12] in the design of wider adders.

In the following, we propose the design of an 8-bit adder module which is composed of two independent carry chains. These chains have the same length (measured as the maximum number of series-connected transistors) as the 4-bit MCC adders. According to our simulation results, the use of the proposed 8-bit adder as the basic block, instead of the 4-bit MCC adder, can lead to high-speed adder implementations.

The derived here carry equations are similar to those for the Ling carries proposed in [15]–[17]. The derived carry equations allow the even carries

to be computed separately of the odd ones. This separation allows the implementation of the carries by two independent 4-bit carry chains; one chain computes the even carries, while the other chain computes the odd carries. In the following, the design of the proposed 8-bit MCC adder is presented.

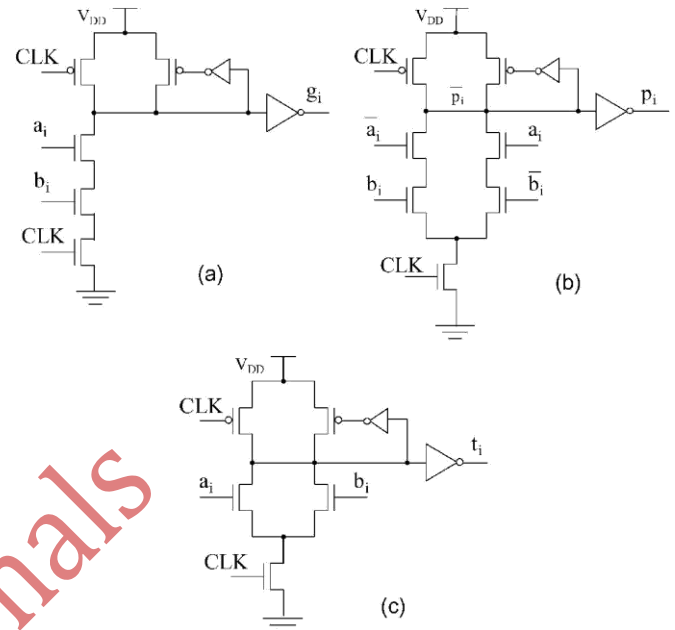


Fig. 1. Domino implementation for the (a) generate, (b) XOR propagate, and (c) OR propagate signals.

#### 3.1 Odd carry computation

The new carries for the odd values of  $i$  are computed according to the aforementioned methodology proposed for the even carries as follows:

$$\begin{aligned} h_1 &= g_1 + g_0 + p_1 p_0 c_{-1} \\ h_3 &= g_3 + g_2 + p_3 p_2 t_1 (g_1 + g_0 + p_1 p_0 c_{-1}) \\ h_5 &= g_5 + g_4 + p_5 p_4 t_3 (g_3 + g_2 + p_3 p_2 t_1 (g_1 + g_0 \\ &+ p_1 p_0 c_{-1})) \\ h_7 &= g_7 + g_6 + p_7 p_6 t_4 \\ &\times (g_5 + g_4 + p_5 p_4 t_3 \\ &\times (g_3 + g_2 + p_3 p_2 t_1 (g_1 + g_0 + p_1 p_0 c_{-1}))). \end{aligned}$$

Let  $G_i = g_i + g_{i-1}$  and  $P_i = p_i \cdot p_{i-1} \cdot t_{i-2}$  be the new generate and propagate signals, respectively, where  $g_{-1} = c_{-1}$

$$\begin{aligned} h_1 &= G_1 + P_1 c_{-1} \\ h_3 &= G_3 + P_3 G_1 + P_3 P_1 c_{-1} \\ h_5 &= G_5 + P_5 G_3 + P_5 P_3 G_1 + P_5 P_3 P_1 c_{-1} \end{aligned}$$

$$h_7 = G_7 + P_7 G_5 + P_7 P_5 G_3 + P_7 P_5 P_3 G_1 + P_7 P_5 P_3 P_1 c_{-1}.$$

### 3.2 Even carry computation

For  $i = 0$  and  $z_0 = t_0$ , from relation (1), we get that  $c_0 = g_0 + t_0 \cdot c_{-1}$ . Since the relation  $g_i = g_i \cdot t_i$  holds, we get that  $c_0 = t_0 \cdot (g_0 + c_{-1}) = t_0 \cdot h_0$ , where  $h_0 = g_0 + c_{-1}$  is the new carry. From relation (2), for  $i = 2$  and  $z_i = p_i$ , we get that  $c_2 = g_2 + p_2g_1 + p_2p_1g_0 + p_2p_1p_0c_{-1}$ .

Since  $g_i + p_i \cdot g_{i-1} = g_i + t_i \cdot g_{i-1}$  and  $p_i = p_i \cdot t_i$ , we have  $c_2 = t_2(g_2 + g_1 + p_2p_1g_0 + p_2p_1p_0c_{-1}) = t_2(g_2 + g_1 + p_2p_1t_0(g_0 + c_{-1})) = t_2 \cdot h_2$

Where  $h_2 = g_2 + g_1 + p_2p_1t_0(g_0 + c_{-1})$  is the new carry.

In the same way, the new carries for  $i = 4, 6$  are computed as  $h_4 = g_4 + g_3 + p_4p_3t_2(g_2 + g_1 + p_2p_1t_0(g_0 + c_{-1}))$   
 $h_6 = g_6 + g_5 + p_6p_5t_4$   
 $\times (g_4 + g_3 + p_4p_3t_2(g_2 + g_1 + p_2p_1t_0(g_0 + c_{-1})))$ .

## 4. PROPOSED SPST

To illustrate the reason of those spurious signal transitions shown in Fig. 1, we explore five cases of 16-bit additions as shown in Fig. 3. The cases of exchanging the operands A and B in additions lead to the same spurious transitions with those shown in Fig. 3. Hence, there is probably no other case beyond these five based on this design. The first case illustrates a transient state in which spurious transitions of carry signals occur in the MSP, although the final result of the MSP is unchanged. Meanwhile, the second and third cases describe situations involving one negative operand adding another positive operand without and with carry-in from the LSP, respectively. Moreover, the fourth and fifth cases demonstrate the addition of two negative operands without and with carry-in from the LSP, respectively. In those cases, the results of MSP are predictable; therefore, the computations in MSP are useless and can be neglected. Eliminating those spurious computations not only can save the power consumption inside the adder/subtractor in the current stage but also can decrease the glitching noises which cause powerwastage inside the arithmetic circuits in the next stage. From the analysis of Fig. 3, we are motivated to propose the SPST that separates the adder/subtractor into two parts and then latches the input data of the MSP whenever they do not affect the computation results. The SPST can be expanded to be a fine-grain scheme in which the adder/subtractor is divided into more than two parts. However, the

hardware complexity of the augmented circuits such as the detection-logic unit, the data latches, and the SE unit increases dramatically. Based on an adder/subtractor example, we actually find that the power expense caused by the augmented circuits is larger than the power reduction in a tripartitioned scheme. This is the reason we propose a bipartitioned SPST scheme in this paper.

### 4.1 Realization Issues of the Proposed SPST

Fig. 5 shows a 16-bit adder/subtractor design example adopting the proposed SPST. In this example, the 16-bit adder/subtractor is divided into MSP and LSP between the eighth and the ninth bits. Latches implemented by simple AND gates are used to control the input data of the MSP. When the MSP is necessary, the input data of MSP remain unchanged. However, when the MSP is negligible, the input data of the MSP become zeros to avoid glitching power consumption. The two operands of the MSP enter the detection-logic unit, except the adder/subtractor, so that the detection-logic unit can decide whether to turn off the MSP or not. Based on the derived Boolean equations (1) to (8), the detection-logic unit of SPST is shown in Fig. 6(a), which can determine whether the input data of MSP should be latched or not. Moreover, we propose the novel glitch-diminishing technique by adding three 1-bit registers to control the assertion of the *close*, *sign*, and *carr-ctrl* signals to further decrease the transient signals occurred in the cascaded circuits which are usually adopted in VLSI architectures designed for multimedia/DSP applications. The timing diagram is shown in Fig. 6(b). A certain amount of delay is used to assert the *close*, *sign*, and *carr-ctrl* signals after the period of data transition which is achieved by controlling the three 1-bit registers at the outputs of the detection-logic unit. Hence, the transients of the detection-logic unit can be filtered out; thus, the data latches shown in Fig. 5 can prevent the glitch signals from flowing into the MSP with tiny cost. The data transient time and the earliest required time of all the inputs are also illustrated in Fig. 6(b). The delay should be set in the range of , which is shown as the shadow area in Fig. 6(b), to filter out the glitch signals as well as to keep the computation results correct. Based on Figs. 5 and 6, the timing issue of the SPST is analyzed as follows.

1) *When the detection-logic unit turns off the MSP:*  
At this moment, the outputs of the MSP are directly compensated by the SE unit; therefore, the time saved from skipping the computations in the MSP circuits shall cancel out the delay caused by the detection-logic unit.

2) *When the detection-logic unit turns on the MSP:*  
The MSP circuits must wait for the notification of the detection-logic unit to turn on the data latches to let the data in. Hence, the delay caused by the detection-logic unit will contribute to the delay of the whole combinational circuitry, i.e., the 16-bit adder/subtractor in this design example.

3) *When the detection-logic unit remains its decision:*

No matter whether the last decision is turning on or turning off the MSP, the delay of the detection logic is negligible because the path of the combinational circuitry (i.e., the 16-bit adder/subtractor in this design example) remains the same. From the analysis earlier, we can know that the total delay is affected only when the detection-logic unit turns on the MSP. However, the detection-logic unit should be a speed-oriented design. When the SPST is applied on combinational circuitries, we should first determine the longest transitions of the interested cross sections of each combinational circuitry, which is a timing characteristic and is also related to the adopted technology. The longest transitions can be obtained from analyzing the timing differences between the earliest arrival and the latest arrival signals of the cross sections of a combinational circuitry. Then, a delay generator similar to the delay line used in the DLL.

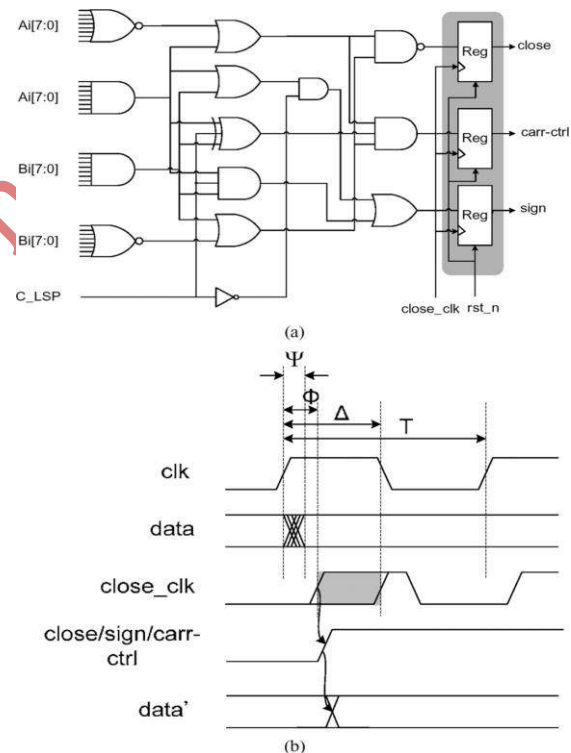
carr-ctrl		$C_{LSP}, A_{and}, A_{nor}$							
		000	001	011	010	100	101	111	110
$B_{and}, B_{nor}$	00	0	0	0	0	0	0	0	0
	01	0	0	0	1	0	1	0	0
	11	0	0	0	0	0	0	0	0
	10	0	1	0	0	0	0	0	1

(a)

sign		$C_{LSP}, A_{and}, A_{nor}$							
		000	001	011	010	100	101	111	110
$B_{and}, B_{nor}$	00	0	0	0	0	0	0	0	0
	01	0	0	0	1	0	0	0	0
	11	0	0	0	0	0	0	0	0
	10	0	1	0	1	0	0	0	1

(b)

**Table 1. Representations of (a) carry-ctrl signal and (b) sign signal in terms of Karnaugh maps.**



**Fig 2: (a) Detection-logic unit and (b) its timing diagram.**

## 4.2 Theoretical analysis and logic derivation

The ETD possesses two types of PEs. From the algorithmic view point, the four PEs in the left-hand side of Fig. 8, denoted by MPE-Is, are used for computing the 1-D transform. Meanwhile, the two PEs on the right-hand side of Fig. 8, denoted by MPE-IIs, are used for computing the 2-D

transform. The ETD possesses a throughput of eight pixels per cycle so that it can perform 720p HD, 1080i HD, and digital cinema video formats at 22, 50, and 100 MHz, respectively (more details of the algorithm derivation and design exploration of ETD can be found in [13]). Based on the previous data analysis, we decide to divide the 15-bit SPST arithmetic units of the MPE-Is into 8-bit MSP and 7-bit LSP and divide the 17-bit SPST arithmetic units of the

MPE-IIs into 8-bit MSP and 9-bit LSP, respectively, as shown in the fraction values near the SPST arithmetic units as shown in Fig. 8 where the SPST arithmetic units are marked with dotted shadows. The adder/subtractors located on the second and the fourth stages of the ETD are kept unchanged because most of the spurious signals existed in the data flowing out of the first and the third stages have been filtered out by the SPST.

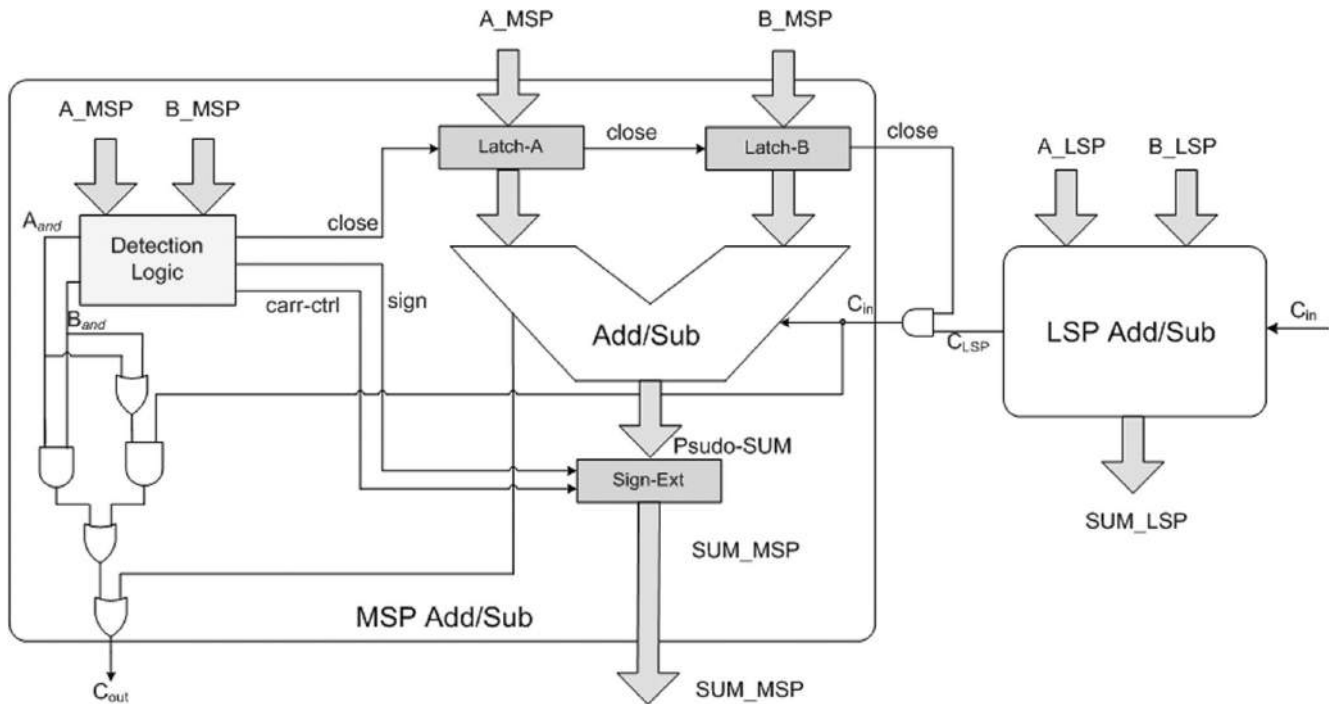
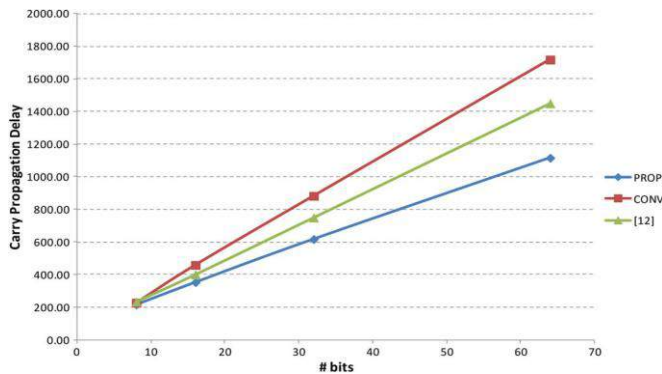


Fig 3: Low-power adder/subtractor design example adopting the proposed SPST

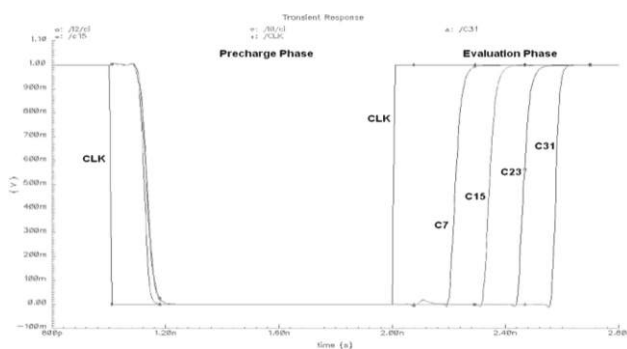
## 5. MCC DESIGN ISSUES AND COMPARISONS

To evaluate the speed performance of the proposed (PROP) design over the conventional (CONV) one, 8-, 16-, 32-, and 64-bit adders have been designed according to the carry chain principle given in Fig. 8(a) and (b), respectively, and simulated using SPECTRE in a standard 90-nm CMOS technology ( $V_{DD} = 1$  V). The conventional 8-, 16-, 32-, and 64-bit MCC adders are designed by cascading two, four, eight, and sixteen 4-bit MCC adder modules. The performance improvements of the PROP design over the CONV design are 23.08% for the 16-bit adder, 30.05% for the 32-bit adder, and 35.08% for the 64-bit adder. Simulated waveforms of the carry signals ( $C_7$ ,  $C_{15}$ ,  $C_{23}$ , and  $C_{31}$ ) for the proposed 32-bit adder are presented in Fig. 9. In all cases previously mentioned, the average energy consumption for a computation is increased by 43.4% for the PROP design with respect to

the CONV one, while the area overhead is 49.9%, due to the extra gates that are required for the implementation of the  $t_i$  and the new generate ( $G_i$ ) and propagate ( $P_i$ ) signals. The proposed technique can be exploited in the design of arithmetic circuits where high performance is required at the expense of power consumption. As referred previously, a modified high-speed design of the 4-bit MCC adder module has been proposed in [12], where the MCC is supported by the carry-skip capability to improve performance. The same technique can also be applied to the chain which computes the odd carries of the proposed



8-bit adder to further improve its efficiency. Since the 8-bit adder is the building block for higher bit adders, in all cases, the performance is proportionally improved. However, even without this addition, the proposed topology outperforms the modified MCC design proposed in [12] since it provides 7.18%, 11.79%, 17.59%, and 23.04% speed improvements for the 8-, 16-, 32-, and 64-bit adders.



## 6. CONCLUSION

The MCC is an efficient and widely accepted design approach to construct CLA adders. Adder fast will be reduced half of the time by increasing bit size. In this way, the speed performance is significantly improved with respect to that of the standard MCC topology. The proposed SPST application verified by skip MSB addition. It proposes an original glitch-diminishing technique to filter out useless switching power we have presented a new Manchester design style that is based on two independent carry chains. Each chain computes, in parallel with the other, half of the carries. In this way, the speed performance is significantly improved with respect to that of the standard MCC topology. The proposed design technique has been applied for the implementation of 8-, 16-, 32-, and 64-bit adders in multioutput domino logic, and the simulation results verified its efficiency.

## 7. REFERENCES

[1] K. Hwang, *Computer Arithmetic: Principles, Architecture, and Design*. New York, NY, USA: Wiley, 1979.

[2] B. Parhami, *Computer Arithmetic, Algorithms and Hardware*. New York, NY, USA: Oxford Univ. Press, 2000.

[3] I. Koren and A. K. Peters, *Computer Arithmetic Algorithms*, 2nd ed. Boca Raton, FL, USA: CRC Press, 2002.

[4] M. D. Ercegovac and T. Lang, *Digital Arithmetic*. San Mateo, CA, USA: Morgan Kaufmann, 2004.

[5] A. Weinberger and J. L. Smith, "A logic for high speed addition," *Nat. Bureau Stand. Circulation*, vol. 591, pp. 3–12, 1958.

[6] J. P. Uyemura, *CMOS Circuit Design*. Boston, MA, USA: Kluwer, 2001.

[7] N. Weste and D. Harris, *CMOS VLSI Design, A Circuit and System Perspective*. Reading, MA, USA: Addison-Wesley, 2004.

[8] P. K. Chan and M. D. F. Schlag, "Analysis and design of CMOS Manchester adders with variable carry-skip," *IEEE Trans. Comput.*, vol. 39, no. 8, pp. 983–992, Aug. 1990.

[9] Z. Wang, G. Jullien, W. Miller, J. Wang, and S. Bizzan, "Fast adders using enhanced multiple-output domino logic," *IEEE J. Solid State Circuits*, vol. 32, no. 2, pp. 206–214, Feb. 1997.

[10] S. Perri, P. Corsonello, F. Pezzimenti, and V. Kantabutra, "Fast and energy-efficient Manchester carry-bypass adders," *Proc. Inst. Elect. Eng.—Circuits Devices Syst.*, vol. 151, no. 6, pp. 497–502, Dec. 2004.

[11] M. Osorio, C. Sampaio, A. Reis, and R. Ribas, "Enhanced 32-bit carry look-ahead adder using multiple output enable-disable CMOS differential logic," in *Proc. 17th Symp. Integr. Circuits Syst. Design*, 2004, pp. 181–185.

[12] A. A. Amin, "Area-efficient high-speed carry chain," *Electron. Lett.*, vol. 43, no. 23, pp. 1258–1260, Nov. 2007.

[13] K. H. Chen, J. I. Guo, and J. S. Wang, "A high-performance direct 2-D transform coding IP design for MPEG-4AVC/H.264," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 4, pp. 472–483, Apr. 2006.

[14] K. H. Chen, K. C. Chao, J. I. Guo, J. S. Wang, and Y. S. Chu, "Design exploration of a spurious power suppression technique (SPST) and its applications," in *Proc. IEEE Asian Solid-State Circuits Conf.*, Hsinchu, Taiwan, Nov. 2005, pp. 341–344.

[15] K. H. Chen, Y. M. Chen, and Y. S. Chu, "A versatile multimedia functional unit design using the spurious power suppression technique," in *Proc. IEEE Asian Solid-State Circuits Conf.*, Hangzhou, China, Nov. 2006, pp. 111–114.

[16] H. H. Chang, S. H. Sun, and S. I. Liu, "A low-jitter and precise multiphase delay-locked loop using shifted averaging VCDL," in *Proc. IEEE Int. Solid-State Circuits Conf.*, Feb. 2003, vol. 1, pp. 434–505.

[17] Y. J. Jung, S. W. Lee, D. Shim, W. Kim, C. Kim, and S. I. Cho, "A dual-loop delay-locked loop using multiple voltage-controlled delay lines," *IEEE J. Solid-State Circuits*, vol. 36, no. 5, pp. 784–791, May 2001.

[18] K. Yano and T. Yamanaka, "A 3.8-ns CMOS 16 × 16-b multiplier using complementary pass-transistor logic," *IEEE J. Solid-State Circuits*, vol. 25, no. 2, pp. 388–395, Apr. 1990.

IJournals