DIFFERENTIAL CASCODE VOLTAGE SWITCH WITH THE PASS–GATE (DCVSPG) LOGIC TREE FOR HIGH PERFORMANCE CMOS DIGITAL SYSTEMS

F. S. Lai and W. Hwang

IBM T. J. Watson Research Center
P. O. Box 218
Yorktown Heights, New York 10598

Abstract

A new circuit configuration, the differential cascode voltage switch with the pass–gate logic tree (DCVSPG), is presented. In this circuit family, we use the pass–gate logic tree to replace the nMOS logic tree in the conventional DCVS circuit in order to eliminate the floating–node problem. By eliminating the floating–node, the DCVSPG shows superior performance, silicon area and power consumption. Moreover, the dynamic DCVSPG also provides the leverage of relieving the charge redistribution concern and reinforces the signal integrity in the typical pre–charge dynamic circuits.

The principle of operation of the DCVSPG is explained. A simple synthesis technique of the pass–gate logic tree is discussed. Finally, a 64–bit carry look–ahead adder is designed by using the static DCVSPG circuit. A nominal cycle time (T_a = 22°C and power supply of 2.5 V) of 2.0 ns is obtained by using a 0.5μm CMOS technology.

I. Introduction

The conventional cascode voltage switch (DCVS) is claimed to have advantages over the traditional static CMOS NAND/NOR design in terms of circuit delay, layout area, logic flexibility and power dissipation [1,2]. For the dynamic implementation, DCVS also shows the superior logic implementation flexibility over the standard domino logic which is suffered from the lacking of inverting gates [3]. However, the conventional DCVS can cause floating–node in both legs of their nMOS logic tree. This floating–node causes the static DCVS to become ratioed logic which in turn creates current spikes and additional delay [4]. Although the ratioed logic problem can be solved in dynamic DCVS by using the pre–charging scheme. Unfortunately, the floating–node problem still exist in the dynamic DCVS, and it will trigger another problem such as the charge redistribution. The result of charge redistribution might develop a false logic evaluation. This makes the dynamic circuit very un–reliable. Complementary pass–transistor logic (CPL) [5] was developed to solve this floating–node problem. The loading in the CPL was chosen using static inverters instead of a cross–coupled pMOS latch in conventional DCVS to restore the signal. This results in a mismatch problem between the input signal level and the logic threshold voltage of the static CMOS inverter. It also caused the CPL to have poorer noise margin and speed degradation. Recently, the double pass–transistor logic (DPL) [6] was developed to solve the CPL problem at the expense of double transistor counts and silicon area. In this paper, a new DCVSPG circuit family is developed [1] to overcome the above mentioned drawbacks in DCVS and CPL. The regeneration problem in DCVS caused by the floating–node is solved by the pass–gate network.

In the following sections, the operation principles of static and dynamic DCVSPG circuits will be presented first. Then the comparison of sum circuits among DCVS, DCVSPG and static CMOS are discussed. Finally, the implementation of a 64–bit carry look–ahead adder by using a 0.5μm CMOS technology will be described. Conclusions will be given in the last section.

II. DCVSPG Circuit Operation Principle

In this section, the basic operation of static and dynamic DCVSPG will be presented. The synthesis of pass–gate logic tree will also be described.

(i) Static Circuit

Fig. 1 shows the typical static DCVSPG to evaluate the AND function of $q = ab$. The *an* and *bn* are the complementary signals of input variables *a* and *b* respectively. Initially, we assume both of *a* and *b* signals are low, the N2 and N4 transistors all turn OFF. However, the *an* and *bn* signals are all high which in turn switch the N1 and N3 transistors ON. It shows that node *q* is low and node *qn* is high. This leads to the cross–coupled pMOS transistor P1 is ON and P2 is OFF. When both of *a* and *b* signals swing from low to high, the node *q* is instantly charged up to high through the N4 transistor. This makes the P1 transistor turn OFF while the node *qn* is discharging through N2 transistor. This is great contrast to the conventional DCVS circuit shown in Fig. 2. In the transition period, the node *q* kept low momentarily which let the P1 transistor ON while the node *qn* is discharging in the conventional DCVS circuit. This floating–node problem causes the conventional DCVS to have larger power consumption and propagation delay. Besides that, for the DCVSPG logic, the pMOS is only used as a load to bring up the full–swing signal, it is not the critical device in the pull–up operation. Therefore, the device size can be small. In the conventional DCVS, however, pMOS has to be twice as large as the nMOS device in order to get a comparable pull–up operation. This leads to a larger silicon area consumption.

Figs. 3 and 4 show the ASTAP simulation results of logic function AND of $q = ab$ for the conventional DCVS and DCVSPG with the function of the pMOS device width. In those simulations, we set the nMOS device width constant (W_n = 20 μm). It is very interesting to note that the rise time is a very strong function of the pMOS width for the DCVS circuit. It is obvious that the rise time decreases when pMOS device width increases. However, the rise time increases again when the pMOS device width increases beyond 10–20 μm. This indicates that the conventional DCVS has the ratioed circuit problem. An optimum pMOS device has to be carefully chosen. On the other hand, the rise time of DCVSPG is almost constant due to the pull–up behavior is mainly done by the nMOS instead of the pMOS. The overall performance of the DCVSPG is much superior to that of the DCVS at any pMOS width and loading.

(ii) Dynamic Circuit

Fig. 5 shows the AND logic function implementation of $q=ab$ for conventional dynamic DCVS and Fig. 6 for the dynamic DCVSPG. When clock signal ϕ is low, both of *q* and *qn* nodes are charged up to V_{dd} high level signal for both of these circuits. Both of circuits start to evaluate the logic function when ϕ is high. For the dynamic DCVS, the node *qn* is starting to discharge when signal *a* and *b* swing from low level 0 to high level V_{dd}. The stored charge in node *qn* will flow through the transistors N3, N4 and N5 to ground. Node *q* is presumably staying in V_{dd} voltage level due to the transistors N1 and N2 turn OFF. However, for the dynamic DCVSPG, the node *qn* is discharged through transistors N2 and N6 to node *bn* which is in the ground state. The node *q* is charged up through transistors N1 and N4 by node *b* which is in the V_{dd} state.

There are several distinct characteristics in DCVSPG in comparison with the conventional DCVS circuit. The advantages are no floating–node generation in both of logic tree legs, symmetrical logic topology and shorter logic stack height. Elimination of the floating–node prevents the static circuit from suffering the ratioed logic problem and the dynamic circuit from the charge redistribution problem. The symmetrical logic topology in the logic tree and the shorter logic stack height improve the circuit performance and power consumption [3].

(iii) Synthesis of Pass–Gate Logic Tree

The synthesis of conventional n–channel logic tree for DCVS had been discussed by using the modified Karnaugh map or the modified Quine–McCluskey tabular method [4]. The synthesis of pass–gate logic had also been explained [7]. However, the synthesis algorithm they showed is either not quite clear or too simple. Let us show a synthesis procedure by using a recursive minimization with the Karnaugh map.

In order to synthesis the logic function $F = \bar{a}\bar{b}\bar{c}\bar{d} + a(b + c + d)$, where $\bar{a}\bar{b}\bar{c}\bar{d}$ are *an, bn, cn, dn* in our figures, into the pass–gate logic, the Karnaugh map result is shown in Fig. 7. The *ab* is assumed to be the control variables and *cd* is the input function variables. By grouping the same output function pattern together, the pass–gate logic can be minimized as $F = \bar{a}\bar{b}(x_1) + b(x_2) + a(x_2)$ as shown in Fig. 7. The x_1 pattern is [1010] and x_2 pattern is [0011]. These two patterns can be continuously minimized as $x_1 = c(d) + \bar{c}(\bar{d})$ and $x_2 = c(1) + \bar{c}(0)$, where 1 is the V_{dd} state and 0 is the ground state. The final pass–gate logic function is then $F = \bar{a}\bar{b}[c(d) + \bar{c}(\bar{d})] + b[c(1) + \bar{c}(0)] + a[c(1) + \bar{c}[0)]$.The DCVSPG circuit is shown in Fig. 7(b). This circuit is much simpler and faster than the pure static CMOS implementation.

III. Comparison of The sum Circuits

In order to compare the circuit performance of various design techniques, the sum circuit of the full adder is being simulated using ASTAP. Fig. 8 shows the static and dynamic versions of the conventional DCVS design technique. The static and dynamic implementation of the DCVSPG is shown in Fig. 9. The static CMOS design of the sum circuit is also shown in Fig. 10.

The device width is designed following a basic rule that the conductance of all the discharging path are assumed to be the same as a conductance of a minimum size ($W_n = 3\mu m$) nMOS transistor. For example, in the conventional DCVS static circuit shown in Fig. 8(a), three transistors are seriesly connected along the discharge path. The transistor width is then chosen as 3 $\mu m \times 3 = 9$ μm. For 8(b), however, the device size is then increased up to 3 $\mu m \times 4 = 12$ μm in the dynamic DCVS configuration. The pMOS device size is chosen as twice larger than that of the nMOS device.

The overall simulation results are shown in Table I. The load output capacitance is assumed that the circuit drives a chain of the similar circuits. With a fan–out of one for the static circuit, it is obvious that the output capacitance is the same with the input capacitance. The dynamic circuits, however, are buffered by the C^2MOS latch and would be expected to be able to drive larger loads than that of the static gate. A fan–out of two was then used for the dynamic designs.

Considering, first, the static design, it appears that the static DCVSPG yields the best power–delay product. The DCVSPG has the lowest logic tree stack height such that its transistor size and input capacitance are the smallest. And yet its best performance is solely due to the pull–up and pull–down are all done by the high–performance nMOS transistor. The lowest power consumption of the static DCVSPG is also due to the very symmetrical charging and discharging times of nodes *q* and *qn* in Fig. 9(a). The asymmetrical charging and discharging periods, however, of the conventional static DCVS causes a prolong transient time when the transistor switches and thus dissipates more power. The DCVS and DCVSPG have the area advantages over the conventional static CMOS circuit due to the redundant pMOS transistors are reduced dramatically in the DCVS configuration.

For the dynamic circuits, it is interesting to note that the power–delay product of the dynamic DCVSPG is the same with the static counterpart in the sum circuit. Of course, its speed is almost twice faster than that of the static DCVSPG at the expense of the larger device count and silicon area.

IV. Static 64–bit Adder Architecture

The whole adder core is shown in Fig. 11. This architecture is implemented by the binary carry look–ahead algorithm [8,9]. There are total 12 rows shown here. The first row is the PG circuit to generate the p (propagation) and g (generate) signals. The last row is the sum circuit. There are total 10 rows to generate the carry signal. Inside the 10 rows of carry chain, the white rectangular and triangular are the buffer circuit and driver circuit respectively. The black rectangular is the merge circuit. Some of white rectangular cell can also route the signal from top to its left to feed into the black merge circuits. The sum bit 0 comes from the right hand side.

From the carry look–ahead theory, the sum bit can be written as

$$s_i = a_i \oplus b_i \oplus c_{i-1} \qquad (1)$$

The generation and propagation bits are defined as

$$g_i = a_i b_i \qquad (2)$$

$$p_i = a_i \oplus b_i \qquad (3)$$

The merge bit can be explained as

$$G_i = g_i + G_{i-1}P_i \qquad (4)$$

$$P_i = p_i P_{i-1} \qquad (5)$$

According to Fig. 11, the merge circuit takes the signals from the top cell and the right–hand signal and outputs signals into the bottom cell. so the g_i and p_i are the signals come from the top cell. However, the G_{i-1} and P_{i-1} are the signals from the right–hand white rectangular cell. By assuming $G_{-1} = c_{-1}$ and $P_{-1} = 0$, we can easily demonstrate that the G_i signal is actually the carry signal c_i with the following definition

$$c_i = g_i + p_i c_{i-1} \qquad (6)$$

With the definition of equations (2) and (3), the PG circuit is shown in Fig. 12. The merge circuit of equations (4) and (5) is shown in Fig. 13. At this circuit, $G_{i-1}n$ is the complementary signal of G_{i-1}. The sum circuit of equation (1) is shown in Fig. 14.

The ASTAP simulation results are shown in Fig. 15 by using a 0.5 μm CMOS technology. All the results are simulated at the nominal condition with $T_a = 22°$C and power supply of 2.5 V. The propagation delay is around 2 ns. The total rows of Fig. 11 in actual ASTAP simulation is 15 stages. This includes 1 driver stage to drive a 0.3 pF capacitive load. This driver stage costs roughly 150 ps delay. Fig. 16 shows the 64–bit adder circuit performance in the function of power supply voltage.

V. Conclusions

The DCVSPG circuit family has been developed. It is shown to have superior performance to that of a conventional DCVS approach. By using the pass–gate logic tree instead of the conventional nMOS logic tree, the floating–node problem is eliminated. This leads to no ratioed logic problem, symmetrical logic topology and shorter logic stack height. A 2 ns 64–bit CMOS adder is achieved by using the static DCVSPG circuit family.

VI.References

[1] F. S. Lai and W. Hwang, to be published
[2] L. G. Heller et. al., Dig. of ISSCC, pp. 16–17, 1984.
[3] L. M. Chu and D. I. Pulfrey, IEEE JSSC, pp. 528–532, 1987.
[4] L. C. Pfennings et. al., IEEE JSSC, pp. 1050–1055, 1985.
[5] K. Yano et. al., IEEE JSSC, pp. 388–395, 1990.
[6] M. Suzuki et. al., Dig. of ISSCC , Paper 5.4, 1993.
[7] D. Radhakrishnan et. al., IEEE JSSC, pp. 531–536, 1985.
[8] R. P. Brent and H. T, Kung, IEEE Comp., pp. 260–264, 1982.
[9] B.W. Wei et. al., Rep. UCB 86/252, UC Berkeley, 1985.

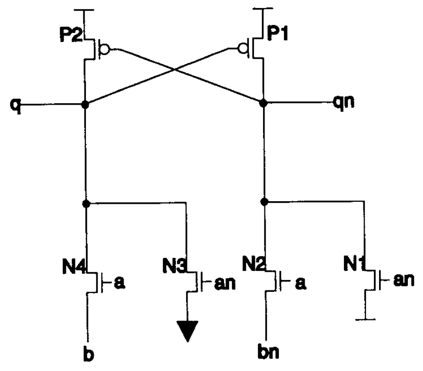

Figure 1. Static DCVSPG

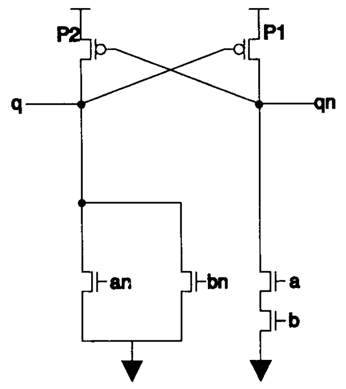

Figure 2. Static DCVS

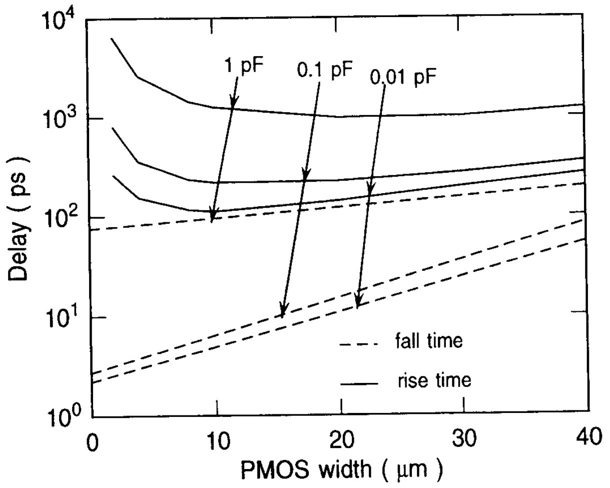

Figure 3. ASTAP simulation results of static DCVS

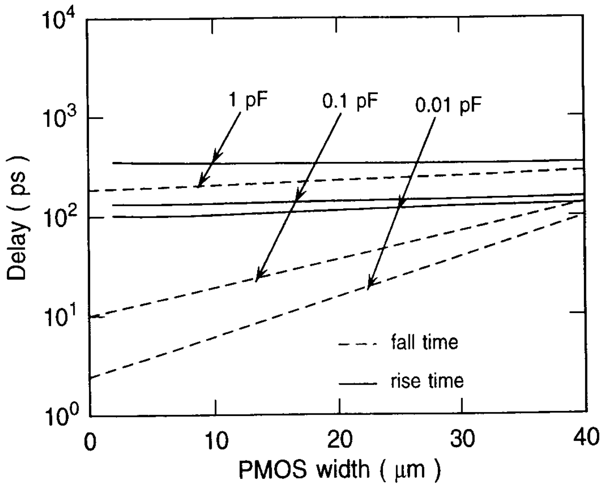

Figure 4. ASTAP simulation results of static DCVSPG

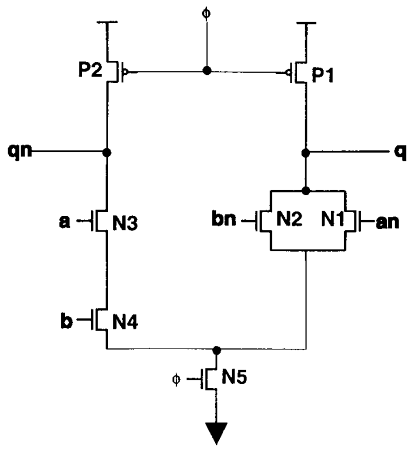

Figure 5. Dynamic DCVS

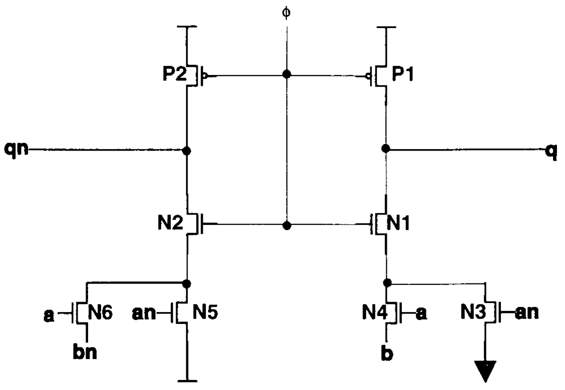

Figure 6. Dynamic DCVSPG

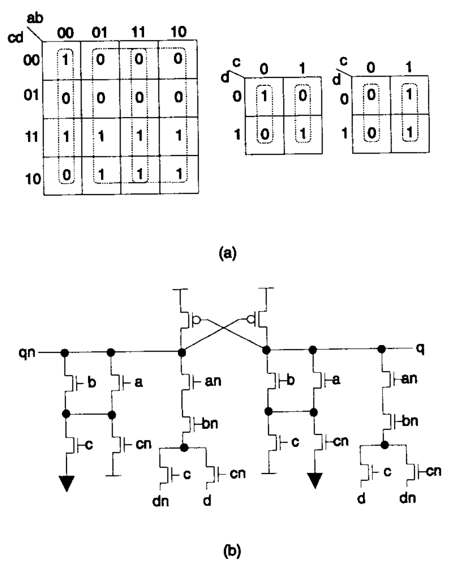

Figure 7. Synthesis of pass–gate logic

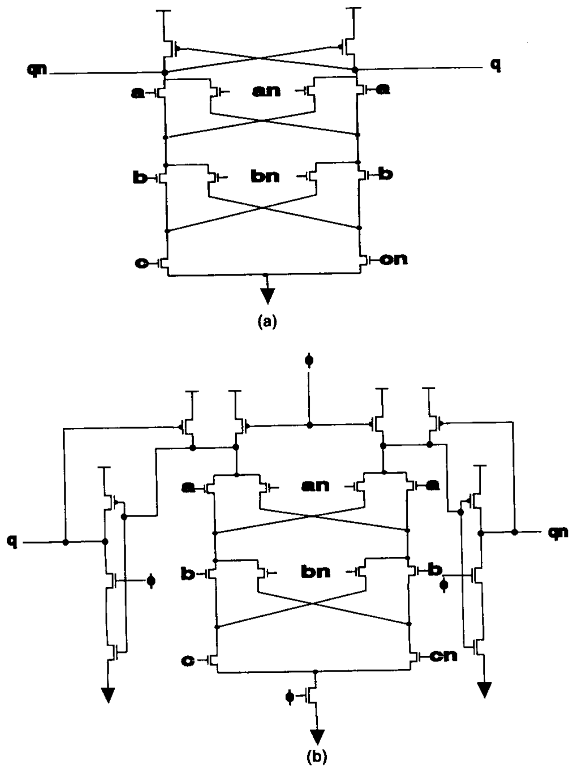

Figure 8. (a) Static DCVS (b) Dynamic DCVS

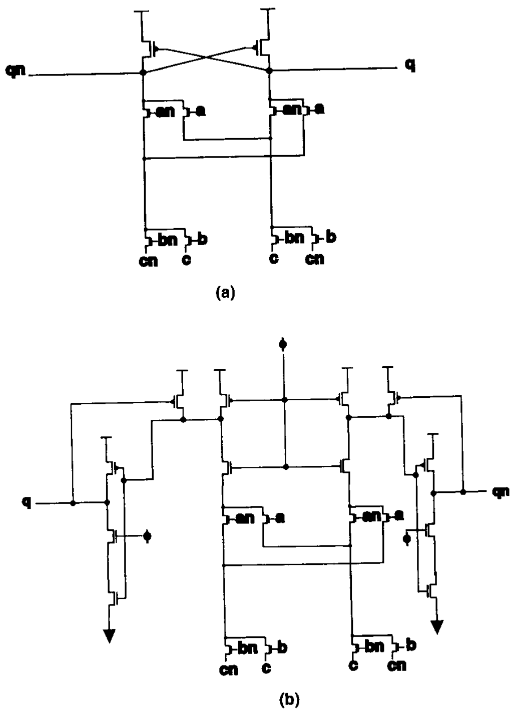

Figure 9. (a) Static DCVSPG (b) Dynamic DCVSPG

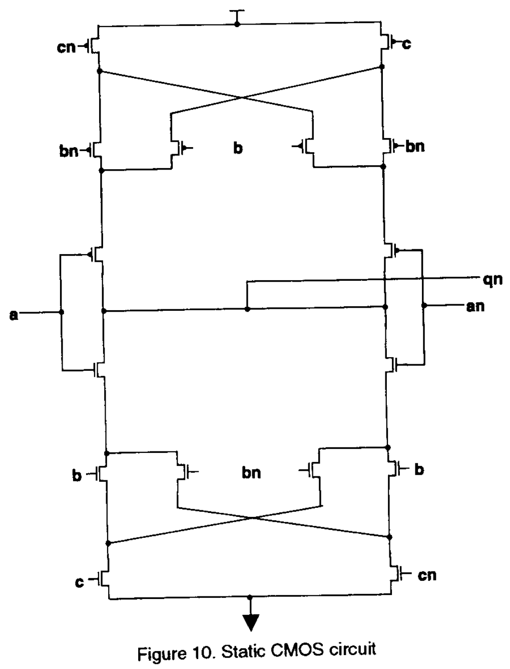

Figure 10. Static CMOS circuit

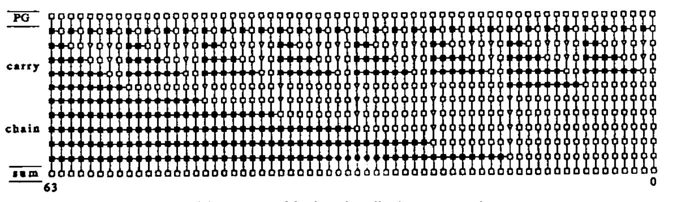

Figure 11. Adder core with the detailed construction

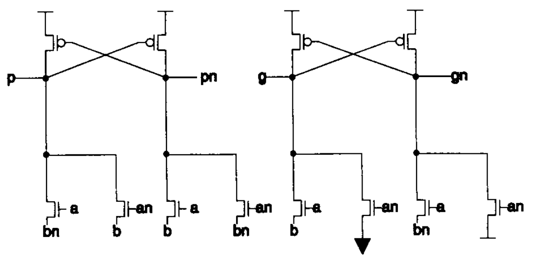

Figure 12. PG circuit implemented with DCVSPG logic family

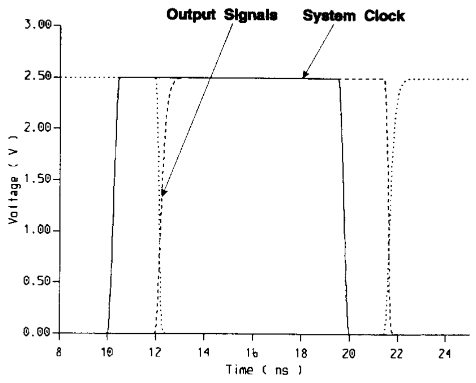

Figure 15. ASTAP simulation results

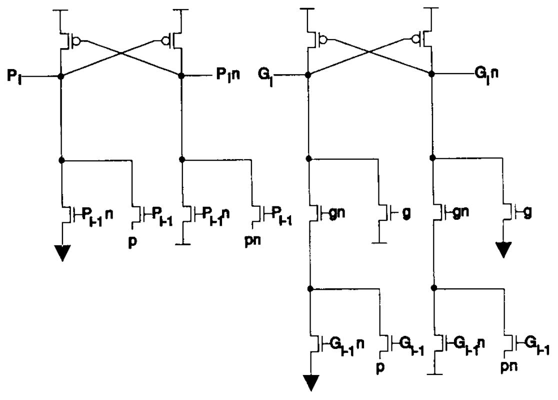

Figure 13. Merge circuit implemented with DCVSPG circuit

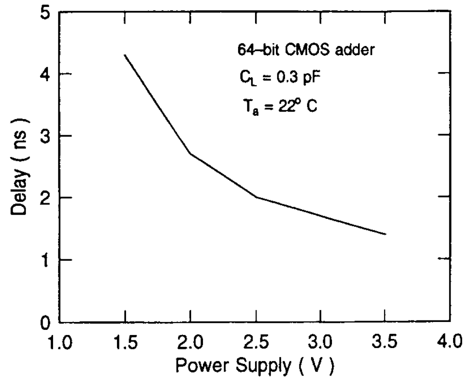

Figure 16. Circuit performance in function of power supply

Table I. Comparison of the sum circuit

	Gate Input Capacitance (fF)	Load Output Capacitance (fF)	P / N	Normalized Area	Delay (ps)	Power (μW)	Normalized Power-Dealy Product
Static CMOS	108	108	8/8	1.00	327	117	1.00
Static DCVS	36	36	2/10	0.62	336	189	1.66
Static DCVSPG	24	24	2/8	0.51	210	48	0.26
Dynamic DCVS	48	96	6/15	1.37	145	118	0.44
Dynamic DCVSPG	36	72	6/14	0.86	122	80	0.26

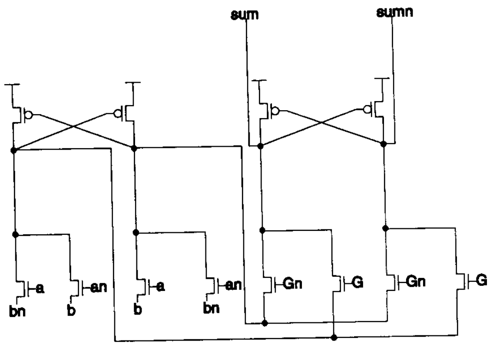

Figure 14. Sum circuit implemented with DCVSPG circuit