Robust High-Performance Low-Power Carry Select Adder

Woopyo Jeong and Kaushik Roy
School of Electrical and Computer Engineering,
Purdue University, West Lafayette, IN 47906, USA
{jeongw, kaushik}@ecn.purdue.edu

Abstract -This paper proposes Dual Transition Skewed Logic (DTSL) based Carry Select Adder (CSA) suitable for processing units requiring low power and high performance with high noise immunity. We implemented 31-bit Carry Select Adders in three different logic styles: Dual Transition Skewed Logic (DTSL), Domino, and conventional static CMOS in TSMC 0.25um technology and compared them in terms of performance, power consumption and layout area. CSA using DTSL shows 36.7% and 17.7% improvements in power dissipation and performance, respectively, over domino, and 40.4% improvement in performance compared to a static CMOS CSA.

I. Introduction

Adders are critical components of the ALU's (Arithmetic Logic Unit) or DSP (Digital Signal Processing) chips. Therefore, high performance adders with low power consumption are essential for the design of high performance processing units. Several different types of high performance adder algorithms are available in literature. Among them, Carry Look–Ahead Adder (CLA) and Carry Select Adder (CSA) are widely used for high speed operations [1][2].

The circuit style is as important as the adder algorithm and architecture. Traditionally, static CMOS circuits have been mostly used in adder designs. However, as the demand for high performance is increasing, several new circuit techniques such as Domino have been used.

However, noise immunity of Domino circuits is worse than static CMOS circuits (especially for scaled technologies) and they consume larger power than static CMOS. Precharge/Evaluate logic using static CMOS technology (e.g. skewed CMOS circuits) is one solution to achieve high performance with low power consumption and good noise immunity [3]. The circuit topology of skewed logic is the same as that of static CMOS logic, however, the PMOS or the NMOS transistors are preferentially sized to achieve fast high-to-low or low-to-high transitions. For example, to speed up high to low transition, the sizes of PMOS transistors are reduced while the NMOS transistors are sized up [4][5].

DTSL (Dual Transition Skewed Logic) is a skewed logic style. However, it does not require clock signal. It has dual paths for data propagation – one path is used for fast propagation of rising transition, while the other path is used for fast propagation of falling transition [5]. In this paper we propose a new CSA using DTSL at the input, which consists of carry propagation logic, control logic, and logic for generating SUM. In order to reduce the carry propagation delay of the proposed CSA using DTSL, we implemented carry propagation logic with one transmission gate and one skewed inverter for each stage. Skewed CMOS circuits are also used in control logic to speed up carry propagation. Logic for generating SUM consists of static CMOS circuits to increase noise immunity and to reduce power consumption.

For comparison we implemented 31-bit CSA using the proposed adder design style, and compared it with 31-bit CSA's using Domino and static CMOS circuits.

The rest of the paper is organized as follows. Section II explains Dual Transition Skewed Logic style. In section III, we explain the standard CSA and CSA using DTSL. Section IV shows implementation and operation of the proposed CSA using DTSL. Sections V and VI show the simulation results and conclusions.

II. Dual Transition Skewed Logic

Skewed logic is suitable for high performance, low power and high noise immunity. However, skewed CMOS circuits do require clock signal, though only a few logic gates may require the clock [4]. Dual Transition Skewed Logic (DTSL) consists of dual data paths using skewed circuits. Fig. 1 shows one example of DTSL circuit. If the input of the first stage of the logic block toggles from high to low, faster data transition takes place through the top data path. On the other hand, if the input toggles from low to high, the data transits faster through the bottom path than through the top path. The arrows represent the skew direction. The combiner detects earliest transition, latches it, and then transfer the data to the next stage [5].

Implementing CSA using DTSL does not require insertion of extra data path since CSA algorithm inherently uses dual data paths; one for input carry of 0, the other for input carry of 1. However, even though there is no extra overhead due to dual data paths, the

The research was sponsored in part by SRC (98-HJ-638), DARPA, and Intel Corporation.

increase in layout area due to control logic for skewed circuits may not be small [5].

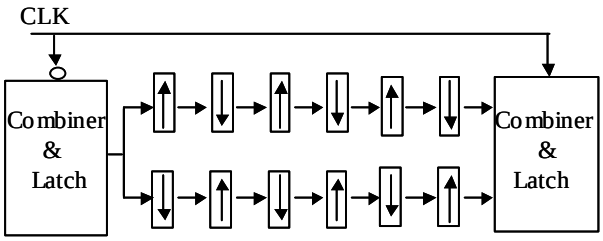

Fig. 1. DTSL block structures

We propose a new CSA using DTSL, which improves performance and has a smaller area than the previous CSA using DTSL. This is achieved by removing carry generation logic on carry propagation paths, and some circuits in control logic.

III. Implementation of Carry Select Adder

A. Carry Select Adder

A general Ripple Carry Adder (RCA) should wait for incoming carry before generating Carry-out for every full-adder cell. However, CSA does not need to do that. Since CSA consists of 2 pairs of logic blocks for carry propagation, and the CARRY inputs for each block are already determined: one is CARRY 0, the other is CARRY 1, each block can generate CARRY outputs simultaneously [6] [7]. Then, the worst delay of the square root CSA having M blocks is determined as

$$t_d = (M-1)t_{mux} + Nt_{carry}$$

where, N is the number of stages in the first block, t_{mux} is delay of multiplexor, and t_{carry} is the carry propagation delay of one stage of ripple carry adder. In order to reduce t_{carry} we should use a fast carry propagation logic style.

B. Carry Select Adder Using DTSL

CSA using DTSL can achieve high performance with robustness and low power consumption comparable to CSA using static CMOS circuits [5]. Fig. 2 shows the block diagram of one *i*th stage of the CSA using DTSL. It consists of logic for generating sum (L7), some complicated control logic (L4, L5, L6), and logic on the carry propagation paths (L1, L2, L3), which is the same as that of a general mirror adder.

In this CSA, Carry-out from the previous stage propagates through the logic on the carry propagation paths when A*k*≠B*k* (for all *k*: $k \le i$), and Carry-out of the current stage is generated by the control logic and switch gates when A*k*=B*k* for any *k* ($k \le i$).

When A*i*=B*i*, the Carry-out of this stage will be generated by control logic (L5, or L6). The output of the first XOR gate, G1 in logic, L7 (logic for generating sum) will be low, which makes the output of G2 low. Therefore, the transmission gates (X, Y) will be turned off. On the other hand, when A*i*≠B*i*, if the inputs of any previous stage are same (A*k*=B*k*, $k<i$) the transmission gates should also be off. For example, in any previous stage (*k*th stage; $k < i$) if A*k*=B*k*, both control signals (CNT<*k*,1>, CNT<*k*,2>) generated by L5 and L6 will be the same (0 or 1) depending on input value. In *k*+1st stage, if $Ak_{+1} \neq Bk_{+1}$, both control signals propagate to the next stage and CNT<*k*+1,1> and CNT<*k*+1,2> will be equal to CNT<*k*,1> and CNT<*k*,2> respectively. Therefore, at the *i*th stage, CNT<*i*-1,1> and CNT<*i*-1,2> will be equal and makes the output of G2 low, which in turn disables the transmission gates.

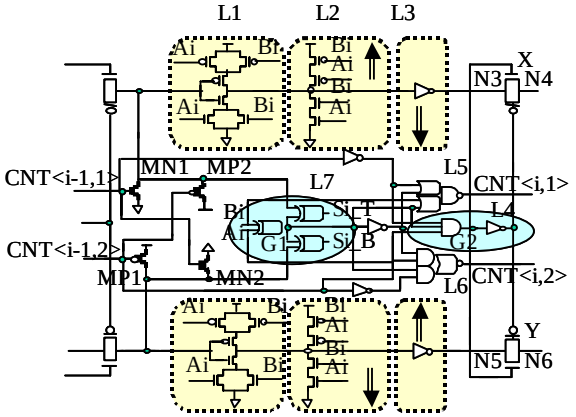

Fig. 2. Block diagram of a stage of CSA using DTSL

We must turn off the transmission gates with the above input conditions to avoid any short circuit current flowing through them. For example, if A*i*≠B*i* and CNT<*i*-1, 1> and CNT<*i*-1, 2> are equal, nodes N3 and N5 will be the same as N4 and N6 respectively. The logic level at N3 and N5 are determined by CNT<*i*-1,1> and CNT<*i*-1,2> and N4 and N6 are determined by CNT<*i*,1> and CNT<*i*,2>. However, since two carry propagation paths have different delays due to the skewing of transistors there may be a shorts circuit current if the transmission gates are not turned off.

The logic for carry generation (L2) on carry propagation paths is used only to maintain input level of L3 when A*i*=B*i*=0 and C*i*=1 or A*i*=B*i*=1 and C*i*=0 because the output of L1 can be floated for these input conditions. The carry propagation logic (L1) propagates Carry-in to the next stage when A*i*≠B*i*. In this case, one PMOS of the top PMOS transistors and one NMOS of the bottom NMOS transistors are always turned on, i.e., L1 acts like an inverter when A*i*≠B*i*. Therefore, L1 can be changed to an inverter having Carry-in as its input, and L2 is not necessary because there is no floating node. This means that the logic of the carry propagation paths of CSA using DTSL can be simplified to two inverters.

IV. Proposed CSA using DTSL

Fig. 3 shows the *i*th stage of the proposed CSA using DTSL. The transmission gates (X, Y) are controlled only by the output of XOR gate G1 of the current stage, i.e., switching of transmission gates is independent of the status of previous stage. In order to reduce the transmission time we use only a skewed inverter between transmission gates on a carry propagation path. However, since only one inverter is used in each stage, the Carry-out of each even stage is opposite to the Carry-out of each odd stage, and hence, the control logic of even stages should be different from that of odd stages. This simplicity of circuits on carry propagation paths reduces the load capacitance, which in turn reduces the critical path delay.

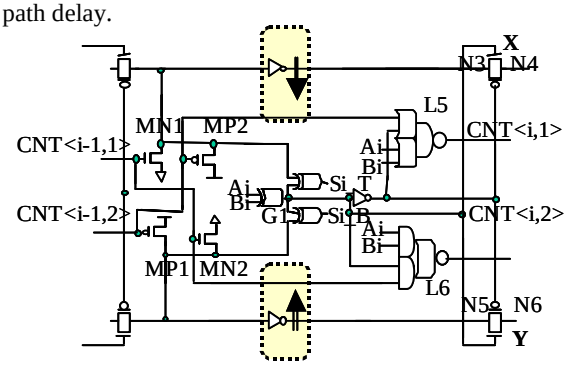

Fig. 3. Block diagram of the proposed CSA using DTSL

Let us explain of operation of the proposed CSA. Fig. 4 shows the CSA consists of two stages. Each stage contains data paths for carry propagation, logic for generating SUM, and control logic. The logic in the circle is for generating the SUM. We can, therefore, improve performance by using properly skewed inverters in the upper and lower carry propagation paths. The skew direction of inverter on the top data path should be opposite to that on the bottom in the same stage.

If inputs A*i*'s are different from B*i*'s (for all $i = 0$ to n), all transmission gates (X, Y) will turn on and the switching transistors (MN*i*, MP*i*) will be disabled. In this case, the operation of the skewed circuits on the carry propagation paths is not different from that of normal static CMOS circuits.

However, the delay time of the skewed circuits is smaller than that of the static CMOS circuits because the direction of CARRY transition in every stage is always the same as the skew direction. Even though transitions of each skewed inverter is fast, the carry propagation delay under this condition is the largest because Carry-out of the first stage will propagate to the last stage. Under such inputs the Carry-outs will be inversions of Carry-in for every stage because each Carry-in goes through one inverter and one transmission gate.

However, if any A*i* is equal to B*i* at Stage *i* ($i = 0$ to n), the Carry-outs on both paths from that stage to the last stage ($i \sim n$) will be the same, and determined only by inputs A*i* and B*i* regardless of Carry-outs of the previous stage. This means that the carry propagation starts simultaneously at the first stage and the *i*th stage.

Hence, in this case, the propagation delay of CSA is the same as one of the carry propagation delays from the first stage to *i*-1st stage and from *i*th stage to the last stage. Then, we have to switch Carry-in of the next stage (*i*+1st) to low or high depending on the value of A*i* and B*i*. For example, let us assume A1=B1=0 at Stage 1, then the outputs of the compound gates, G1_T and G1_B, in the control logic of this stage will be 0, and PMOS switching transistor (MP1) will turn on. Therefore, the Carry-in (C2_T, C2_B) at the next stage will be low regardless of Carry-in (C1_T, C1_B) of Stage 1. Hence, we do not need to wait for the Carry-in to propagate to the output node of Stage 1, i.e. when inputs A1, B1 of Stage 1 are set, we can switch Carry-in of the next stage (Stage 2) immediately to low after turning off the transmission gates on data path. Similarly, if A1=B1=1 at Stage 1, then we change Carry-in of the next stage (Stage 2) to high. For such cases, the total propagation delay will be shorter than the total delay of the previous case ($Ai \neq Bi$, for all $i = 0$ to n) because the time taken to switch Carry-in of the next stage (Stage 2) is shorter than the time in which Carry-in of the first stage (Stage 0) propagates to the Carry-in node of Stage 2 having A2, B2 as inputs.

Finally, let us consider the case when $Ai = Bi$ and $Aj \neq Bj$ ($j=i+1$). In this case, even though A*j* is not the same as B*j*, we have to switch Carry-in of *j*+1st stage on the data path because the skew direction of one data path at Stage (*j*+i) is opposite to Carry-in of *j*+1st stage. For example, when A1=B1=0 at Stage 1 and A2=0, B2=1 at Stage 2, the outputs of the compound gates (G1_T, G1_B) in the control logic of stage 1 will be 0 and PMOS switching transistor (MP1) will turn on, which makes Carry-in of Stage 2 (C1_T and C1_B) low. In this condition, although C1_T switches fast from high to low, node C1_B switches to low slowly because the transition direction is opposite to the skew direction (arrow direction). At the next stage (Stage 2) since A2=0 and B2=1, the transmission gates are on, and data on C1_T and C1_B should be propagated to C2_T and C2_B. However, the delay of the bottom data path increases because the transition direction is opposite to the skew direction. Therefore, at the final stage the delay of the bottom data path may be very large. In order to prevent this effect, the control logic at Stage 2 generates Carry-in for the next stage. The compound gate (G2_T) in the control logic of Stage 2 has 4 inputs: one is the output of G1_B, and others are A2, B2, and the output of XOR gate G2.

Since A2=0, B2=1, the output of the XOR gate G2 is high, and the output of G1_B is also low. Therefore, the output of G2_T is high, which makes NMOS switching transistor (MN2) turn on, and the Carry-out (C3_T, C3_B) at that stage be low.

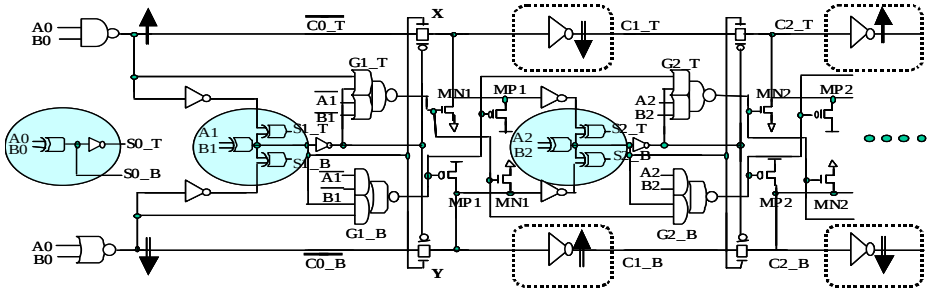

Fig. 4. Block diagram of the proposed Carry Select Adder

a) 31-bit Carry Select Adder using Domino circuits

b) 31-bit Carry Select Adder using Static CMOS circuits

c) 31-bit proposed Carry Select Adder using DTSL

Fig. 5. Layouts of 31-bit CSA in different styles

V. Results

To compare the propagation delay, the power consumption and layout area of the proposed CSA with CSA using Domino logic and static CMOS logic, we implemented 31-bit CSA's having 5 blocks, laid out, and simulated each design in TSMC 0.25µm CMOS technology with Vdd=2.5 volts.

Table 1 shows comparison of simulation results and layout area of each CSA. Fig. 5 shows the layouts of the proposed CSA using DTSL and CSA's using Domino and static CMOS circuits. The propagation delays are obtained under the worst case carry propagation. The average power consumption is obtained with random input vectors with a clock cycle of 10ns.

Table 1 shows that the proposed adder is superior to static adder in terms of performance. Comparison shows that the proposed CSA has a 36.7% power improvement with 17.7% improvement in performance compared to Domino CSA. The results also show that the area of the proposed CSA is comparable to that of the static CMOS implementation. The power delay product of the proposed CSA is almost half of that of Domino CSA.

VI. Conclusions

In this paper we proposed a robust high performance low power carry select adder using DTSL and transmission gates. Results show that the proposed 31bit Carry Select Adder has superior performance to Domino based CSA implementation while being 36.7% more power efficient. The layout area is comparable to static CMOS CSA.

References

[1] T. Lynch, et al., "A Spanning Tree Carry Lookahead Adder," IEEE Trans. On Computers, vol. 41, no. 8, pp. 931-939, Aug., 1992.

[2] V. Kantabutra, "A Recursive Carry-Lookahead / Carry-Select Hybrid Adder," IEEE Trans. On Computers, vol. 42, no. 12, pp. 1495-1499, Dec., 1992.

[3] D. Somasekhar, "Power and dynamic noise considerations in high performance CMOS VLSI design," *PhD Thesis*, Purdue Univ., Aug., 1999.

[4] A. Solomatmikov, D. Somasekhar, K. Roy, "Skewed CMOS: Noise-immune high-performance low-power static circuits family," *ESSCIRC*, 2000.

[5] W. Jeong, K. Roy, and C. Koh, "High-Performance Low-Power Carry Select Adder using Dual Transition Skewed Logic," *ESSCIRC*, 2001.

[6] N. Weste and K. Eshraghian, "Principles of CMON VLSI Design", Addison Wesley Publishers, 1994.

[7] O. Bedrij, "Carry-Select Adder," *IRE Trans. on Electronic Computers*, Vol. EC-11, pp.340-346, 1962.

TABLE 1
Comparison results for 31-bit Carry Select Adders

Parameter	Domino Logic	Static CMOS	Proposed Adder	Improvement over Domino	Improvement over Static
Delay [ns]	1.812	2.501	1.491	17.7 %	40.4%
Power [mW]	5.092	2.667	3.224	36.7 %	-20.9%
PDP [mW X ns]	9.227	6.670	4.807	47.9 %	27.9%
Layout Area	24,090	28,200	29,727	-23.3 %	-5.4%