470ps 64bit Parallel Binary Adder*

Jaehong Park[1], Hung C. Ngo[2], Joel A. Silberman[2], Sang H. Dhong[2]

[1]Samsung Electronics, San #24 Nongseo-Ri, Kiheung-Eup, Yongin-City, Kyunggi-Do, Korea 449-711
[2]Austin Research Laboratory, IBM, 11400 Burnet Road, MS 9460, Austin, TX 78758

Abstract

This paper presents a fast 64-bit parallel carry look-ahead binary adder implemented in a 1 GHz research prototype 64-bit PowerPC microprocessor. Efficient use of dynamic compound gates enables implementation of the adder in just three stages of delayed reset dynamic logic. The computation uses only G (Generate) and P (Propagate), and the inverse of Carry is computed from G, P, and a strobe signal.

Introduction

A fast and efficient adder is important in CPU design. Addition latency is often in the critical path of high performance microprocessors. A number of adder architectures have been proposed and implemented in various circuit design styles. As performance targets become more aggressive, parallel adder architectures implemented in dynamic circuit design style become more popular [1][2].

This paper presents a fast parallel 64-bit carry look-ahead adder implemented in delayed-reset dynamic circuit design style [3]. Its carry computation is similar to prefix computation of [4]. The adder has been realized in a 1GHz research prototype 64-bit PowerPC microprocessor [5]. Its circuit simulation delay is 470ps in 0.12/0.15 (n/p)μm L_{eff}, (0.225μm L_{drawn}), 1.8V Vdd, IBM CMOS 7S bulk technology with copper interconnect. The simulation condition is 1.62V Vdd and 85°C.

Adder Architecture and Design

Our adder is composed of 3 stages of delayed-reset dynamic circuits. Figure 1 shows the adder architecture. Figure 2 shows 4-way merging of Generate and Propagate signals. Only a couple of highest-order and lowest-order bit signals are shown for brevity. Although not shown, for example, inputs to the bit 48 of the second stage come from bits 48, 52, 56, and 60 of the first stage, and it fans out to bits 0, 16, 32, and 48 of the third stage. Figure 3 shows timing of signals in Figure 1.

Each stage in Figure 1 merges G (Generate) and P (Propagate) by 4-bits. At the first stage, G4 and P4 are computed. Suppose A and B are data inputs to the adder. G (Generate) and P (Propagate) at i-th bit are computed as:

$G_i = A_i \cdot B_i$ --- (eq. 1)

$P_i = A_i + B_I$ --- (eq. 2)

In previous adder designs [2], G4 (a group of 4 Generate) at *i-th* bit is computed based on the following:

$G4_i = G_i + P_i \cdot G_{i+1} + P_i \cdot P_{i+1} \cdot G_{i+2} + P_i \cdot P_{i+1} \cdot P_{i+2} \cdot G_{i+3}$ --- (eq. 3)

After substituting (eq. 1) and (eq. 2) in (eq. 3), implementing (eq. 3) with a dynamic gate needs to use 5 Nfets in series. However, from (eq. 1) and (eq. 2) we can deduce:

$G_i \cdot P_i = G_i$ --- (eq. 4)

Using (eq. 4), (eq. 3) is simplified into the following:

$G4_i = (G_i + G_{i+1} + G_{i+2} + P_{i+2} \cdot G_{i+3}) \cdot (G_i + P_i \cdot P_{i+1})$ ---(eq. 5)

(Eq. 5) can be implemented as in Figure 4. While the critical path of the dynamic gate in (eq. 3) is 5 Nfets and 1 Pfet, the critical path of the compound dynamic gate in Figure 3 is 3 Nfets and 2 Pfets. Simulation results showed that a compound dynamic gate as in Figure 3 was about 15-20% faster than a dynamic gate implementing (eq. 3).

The simple form of the compound gate in Figure 3 can be used in the second and third stages of the adder if the property that P includes G is maintained. Suppose $i \le k \le n$, $G_{i...k}$ and $P_{i...k}$ are Generate and Propagate signals of higher order bits respectively, and $G_{k+1...n}$ is Generate signal of lower order bits. Then, simplified equation of Carry is

$$\begin{aligned} Carry &= G_{i...k} + P_i \cdot P_{i+1} \cdots P_k \cdot G_{k+1...n} \\ &= G_{i...k} + (P_i \cdot P_{i+1} \cdots P_k + \varphi) \cdot G_{k+1...n} \end{aligned}$$ --- (eq. 6)

for any $\varphi \subseteq G_{i...k}$.

Considering (eq. 6), P4 is computed such that the relation of P and G in (eq. 4) also holds for P4 and G4, and thus enables implementation of G16 with a dynamic compound gate as in (eq. 5). Thus,

$$\begin{aligned} P4_i &= (G_i + G_{i+1} + G_{i+2} + P_{i+2} \cdot P_{i+3}) \cdot (G_i + P_i \cdot P_{i+1}) \\ &= P_i \cdot P_{i+1} \cdot P_{i+2} \cdot P_{i+3} + \varphi \end{aligned}$$ --- (eq. 7)

where

$\varphi = (G_i + G_{i+1} + G_{i+2}) \cdot (G_i + P_i \cdot P_{i+1})$ and $\varphi \subseteq G4_i$.

Implementation of P4 in the form of (eq. 7) increases load to input signals. However, simulation results showed that the advantage of using a compound dynamic gate outweighs disadvantage of increased load for speed improvement. G16 and P16 are computed in a similar way to G4 and P4 respectively.

At the third stage of Figure 1, SUM and COUT are computed. Figure 5 shows the circuit schematic to compute SUM of bit 0 to bit 14:

$$\begin{aligned} SUM_i &= HS_i \oplus Carry_i \\ &= HS_i \oplus ((G16_{i+1} + G16_{i+17} + G16_{i+33} + P16_{i+33} \cdot G16_{i+49}) \cdot \\ &\quad (G16_{i+1} + P16_{i+1} \cdot P16_{i+17})) \end{aligned}$$

Since the inverse of $Carry_i$ is needed in the XOR to compute the SUM and only the positive sense of $Carry_i$ is computed in this design, a timed strobe signal is used in the last stage to invert $Carry_i$. Consider Figure 5. Since nets *dyn1* and *dyn2* are precharged high, *Strobe* signal is timed such that it does not arrive before the worst case timing of nets *dyn1* or *dyn2* with some margin. *Strobe* signal is generated from data inputs of the adder or clock signal. Pmos transistors *P1* and *P2* also provides protection against premature evaluation. If net *dyn_hs_cnot* falls because *strobe* is applied before the eventual fall of *dyn1* or *dyn2*, *P1* and *P2* will pull up after short period of time minimizing the amount of glitch.

Computing only P and G result in fewer wires and devices, and resultantly less power, compared to a dual rail design that directly computes both $Carry_i$ and $\hat{}Carry_i$. In addition, P and G have considerably less signal activity than P, G, and Z if probability of 1's is much less than probability of 0's at the adder inputs. In some adder applications, CPU instructions often activate only lower-order input bits as in the case of adding bytes or half words. Thus, computing only P and G can be considerably more power efficient than computing all of P,G, and Z.

* This work was done when the author[1] was with IBM.

 0-7803-6309-4/00/$10.00 © 2000 IEEE

Experiment Results

The adder was implemented in a 1GHz research prototype PowerPC microprocessor [5] using 1.8V, 0.12/0.15(n/p)μm L_{eff} (0.225μm L_{drawn}) IBM CMOS7S bulk technology. Figure 5 shows the actual layout. The adder occupies 190μm x 806μm, and its transistor count is 12846. The speed of a microprocessor test chip with the adder in the critical path was measured at 1.15 GHz (1.8V Vdd), which verified the adder speed.

Conclusion

A 64-bit parallel carry look-ahead binary adder was designed and implemented in a 1GHz research prototype PowerPC microprocessor. It demonstrated high-speed operation. The adder is composed of three stages of delayed-reset dynamic gates, and makes efficient use of compound dynamic gates to achieve high speed.

References

[1] W. Hwang, et al., "Implementation of a Self-Reseting CMOS 64-Bit Parallel Adder with Enhanced Testability," IEEE JSSC, vol. 34, pp. 1108-1117, Aug. 1999.

[2] S. Naffziger, "A Sub-Nanosecond 0.5μm 64b Adder Design," ISSCC Tech. Dig. pp.362-363, 1996.

[3] J. Silberman, et al., "A 1.0 GHz Single-Issue 64-Bit PowerPC Integer Processor," IEEE JSSC, vol. 33, pp. 1600-1608, Nov. 1998.

[4] P. Kogge and H. Stone, "A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations," IEEE Trans. on Computes, vol. C-22, pp. 786-792, March 1982.

[5] P. Hofstee, et al., "A 1 GHz Single-Issue 64b PowerPC Processor,", to appear in ISSCC Tech. Dig. 2000.

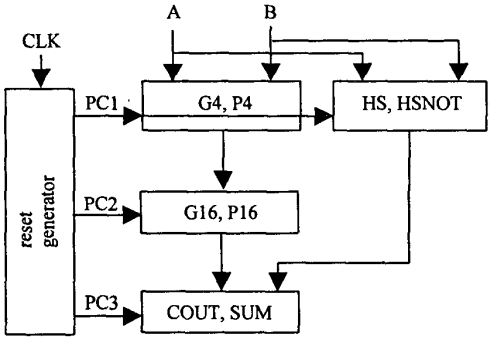

Fig. 1. Block diagram of the adder

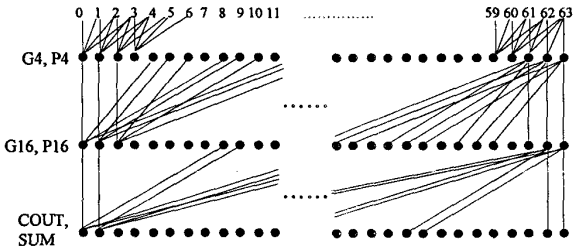

Fig. 2. 4-way merging of G and P

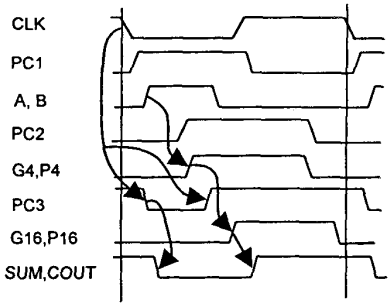

Fig. 3. Timing diagram of signals in Fig. 1

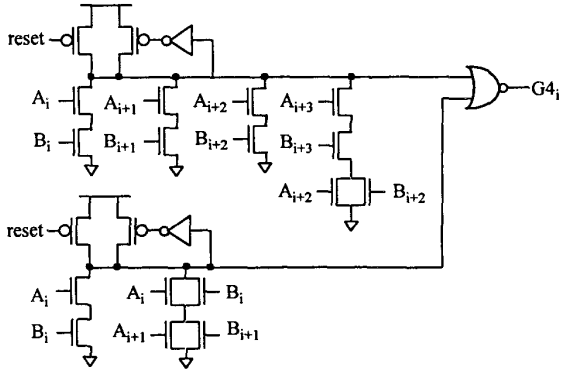

Fig. 4. Compound dynamic gate to compute G4

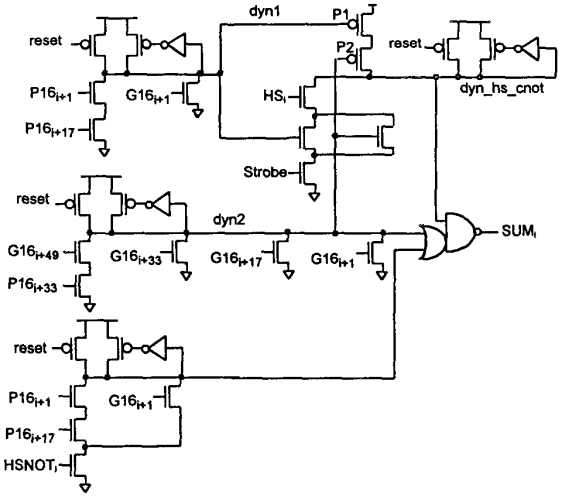

Fig. 5. Compound dynamic gate to compute SUM

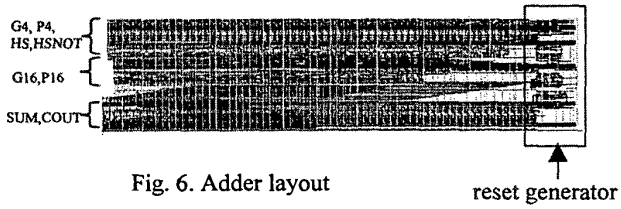

Fig. 6. Adder layout