A 372ps 64-bit Adder using Fast Pull-up Logic in 0.18-μm CMOS

Jooyoung Kim, Kangmin Lee and Hoi-Jun Yoo
Dept. of EECS, Korea Advanced Institute of Science and Technology(KAIST)
373-1, Guseong-dong, Yuseong-gu, Daejeon, 305-701, Republic of Korea
trample7@eeinfo.kaist.ac.kr

***Abstract*—This paper presents a 372ps 64-bit adder using Fast Pull-up Logic (FPL) in 0.18 μm CMOS technology. Fast Pull-up Logic is devised and applied to decrease pull-up time which is critical in domino-static adder. The implemented adder measures the worst case delay of 372ps. The adder has a modified tree architecture using Load Distribution Method and has 6 logic stages.**

I. Introduction

A 32-bit or 64-bit adder is one of the essential arithmetic units of microprocessors. For the adders, computing speed is the most critical issue because the performance of the microprocessor is directly affected by the arithmetic units. Novel logics and fast 64-bit adders have been actively researched by many groups including ours [1]-[4]. Recently, designs using multiple stages of domino-static logic are frequently used for high-speed of adders [5]. However, in the domino-static logic, pull-up delay time of the static stage is much longer than pull-down delay of the dynamic stage, and this limits speed of the adder. To improve the pull-up delay of the static logic, we propose a new novel logic named Fast Pull-up Logic (FPL) that can be applied to generate group G signals, and to replace the static logic. In this paper, we present a 64-bit adder using the FPL. Its worst case delay is reduced from 540ps to 372ps due to the FPL, which is the shortest delay among the 64-bit adders reported in the same process technology. We also propose a load distribution method (LDM) to improve the adder performance further by distributing fan-out loading on each tree stage.

The remainder of this paper is organized as followings. In Section II, the FPL is proposed and evaluated. Section III will describe the modified tree architecture designed by LDM. How to apply FPL to this architecture is also included. In Section IV, simulation results of the implemented 64-bit adder and comparison with other 64-bit adders are presented. Finally, we will conclude and summarize our paper in Section V.

II. Fast Pull-up Logic for Group G Generation

A. Motivations

Recently tree structured carry-lookahead adder with domino-static logic has been popularly used for high performance adders [5]. Domino-static logic uses dynamic and static logic alternately by stage and its worst case delay is occurred when outputs of dynamic stages make full transitions from 1 to 0 while outputs of static stages switch from 0 to 1 as depicted in Fig. 1(a). However, if we examine the waveform at the worst case in Fig. 1(b), pull-up delay times of the static stages are about 30ps longer than pull-down delay times of dynamic stages. This is due to the lower conductivity of PMOS transistors compared with NMOS transistors even if the PMOS transistors are sized 2 or 3 times bigger. Moreover, if the PMOS transistors are connected in series in pull up network (PUN), the channel width should be larger to compensate the delay degradation. But the large sized PMOS transistor offers large loading to its previous stage and this causes lag of pull-down delay time further. To solve this limitation of the domino-static logic, we propose the Fast Pull-up Logic (FPL) which can make pull-up process as fast as dynamic stage's pull-down process and offers small loading to its previous stage.

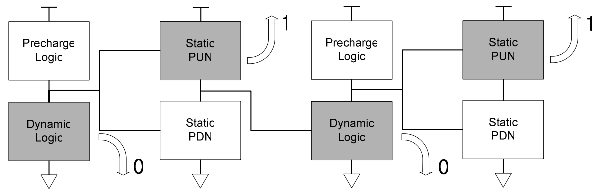

Figure 1(a). Worst case behavior of domino-static logic

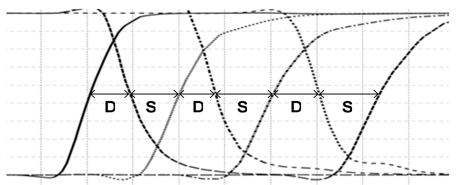

Figure 1(b). Waveform of worst case in domino-static logic

B. Group G generation using FPL

Carry-lookahead adder generates the carries in parallel by using the equation $C_i = G_i + P_iG_{i-1} + P_iP_{i-1}G_{i-2} + \ldots$, where $G_i = A_i \bullet B_i$ and $P_i = A_i + B_i$ [6]. In general, the logarithmic carry merging tree structure is used to compute C_i. The first dynamic stage generates G_i and P_i from input vectors $/A_i$ and $/B_i$, and second static stage starts carry merging by generating two signals *group generate* GG ($GG_{i:i-1} = G_i+P_i \bullet G_{i-1}$) and *group propagate* GP ($GP_{i:i-1} = P_i \bullet P_{i-1}$). Therefore, in practice, static stages produce inverting of GG and GP through /(A+B•C) and /(A•B) operation while dynamic stages produce group GG and GP through /(A·(B+C)) and /(A+B) operation, respectively (Fig. 2). In this case, critical path is caused

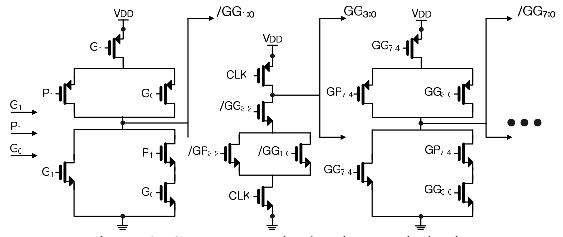

Figure 2. Carry merge in domino-static logic

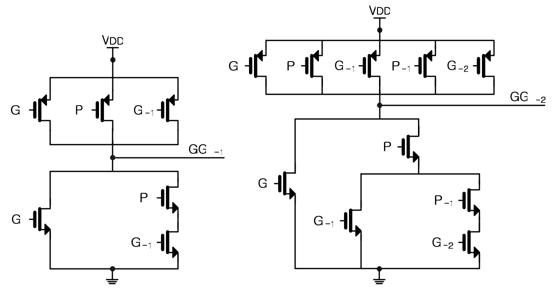

Figure 3. Circuits of group G generation in FPL

by GG generation of static stage because it is relatively slower than GP generation. To decrease pull-up time of GG generation in static stage, we propose Fast Pull-up Logic for 2-input and 3-input GG generations ($GG_{i:i-2} = G_i + P_i\bullet(G_{i-1}+P_{i-1}\bullet G_{i-2})$) as shown in Fig. 3

The FPL logic for 2-input GG generation is shown again in Fig. 4 with its previous dynamic stage. The NMOS pull-down circuit is the same as static pull-down logic but FPL's PMOS circuit is connected in parallel regardless of its functional logic. Therefore output logic is determined only by NMOSs and the role of PMOSs is specified to offer a strong pull-up driving force. This is the main difference between static logic and the FPL. Detailed operations of FPL and how it achieves fast pull-up process will be described in the next.

1) When the clock is 0, dynamic stage precharges output voltage up to V_{DD}. Therefore, all inputs of the FPL become 1 and output of the FPL tied to 0 in precharge phase.

2) When the clock is 1, FPL evaluates logic after previous dynamic stage finishes its transitions. Because pre-condition of FPL's output is 0, either going up to V_{DD} or keeping 0 is possible in evaluation phase. In the pull-up process, the pull-down network is turned off (*i.e.* N1 and N2 or N3 are turned off.) At the same time, P1 and P2 or P3 are turned on. That means more than 2 PMOSs are always turned on for the pull-up process. Because multiple PMOSs connected in parallel reinforce each other, FPL can have strong pull-up driving force with small widths. In other case, if the logic is 0, the current path from V_{DD} to ground is generated except when all inputs are 1, and output voltage V_{OL} comes out in ratioed value of equivalent resistance of pull-up and pull down network. Therefore, keeping V_{OL} under certain level and getting strong pull-up driving force are issues of FPL. If we fix the size of NMOSs to W_n, rising time of the FPL t_r and V_{OL} are both dependent to PMOSs' width. The PMOS widths are determined by the following equations on t_r and V_{OL} on assumption that width of P1 is W_{p1}, width of P2, P3 are same as W_{p2}, and W_p is the sum of W_{p1} and W_{p2}.

Assume that load capacitance of output is C_L, voltage of output is V_C and current of C_L is I_C, $\Delta Q = Ic\cdot\Delta t = C_L\cdot\Delta V_C$. Then the rise time t_r is given as (1),

$$t_r = \frac{1.6L\cdot C_L\cdot V_{DD}}{\mu_P C_{ox}(V_{DD}-V_T)^2}\times\frac{1}{W_p}=\frac{k}{W_p}, \quad k=\text{constant} \qquad (1)$$

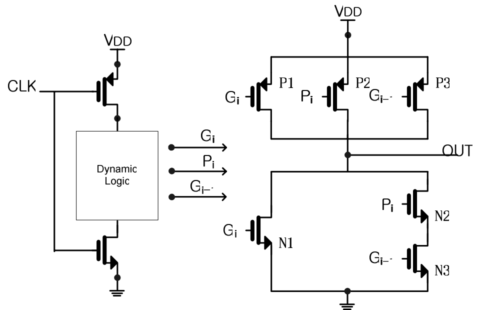

Figure 4. Circuit of 2-input group G generation in FPL

TABLE I. OPTIMIZED SIZES OF FPL SHOWN IN FIG.4.

Transistor	**P1**	**P2**	**P3**
Width/length(μm)	2.4 / 0.18	1.2 / 0.18	1.2 / 0.18
Transistor	**N1**	**N2**	**N3**
Width/length(μm)	1.5 / 0.18	3.0 / 0.18	3.0 / 0.18

If the equivalent resistance of unit length of NMOS is R_n and PMOS is R_p, respectively, the maximum of V_{OL} is occurred when P1, N2 and N3 are turned on or P2, P3 and N1 are turned on. Therefore V_{OL} will be,

$$V_{OL}=Max\left\{\frac{(R_n/W_n)}{(R_p/W_{p1})+(R_n/W_n)}V_{DD},\ \frac{(R_n/W_n)}{(R_p/2W_{p2})+(R_n/W_n)}V_{DD}\right\} \qquad (2)$$

If we set V_{OL} to $r\cdot V_{DD}$, we can get the optimized W_{p1} and W_{p2} value as given by (3).

$$W_{p1}=\alpha\cdot\left(\frac{W_nR_p}{R_n}\right)\cdot\left(\frac{r}{1-r}\right),\quad W_{p2}=\frac{\alpha}{2}\cdot\left(\frac{W_nR_p}{R_n}\right)\cdot\left(\frac{r}{1-r}\right) \qquad (3)$$

r value is limited by the noise margin of the next stage so it must not exceed 0.25 (*i.e.* V_T/V_{DD}), practically. And W_n value is limited by loading effects to previous stage. In (3), we compensate the approximation error of device nonlinearity by multiplying the constant α which is about 2. The optimized sizing results of 2-input GG generation circuit in FPL are listed in Table I. This optimized FPL circuit always guarantees pull-up driving force as much as a 3.6-mm width PMOS. In static logic of GG generation, 3 PMOSs whose widths are 7.2-mm are required to put out same driving power. FPL decreases the sum of PMOS widths by 77.8% under same driving power condition. 3-input GG generation circuit can be optimized using the same method.

C. Performance Analysis

We simulate FPL with previous dynamic stage to consider the

TABLE II. THE SIMULATION RESULTS IN 2-INPUT GG GENERATION.

2-input GG generation	**Dynamic + static**	**Dynamic + FPL**
Pull-up delay (ps)	124	96
V_{OL} (mV)	0	280
Sum of logic TR sizes (μm)	22.5	12.3

3-input GG generation	**Dynamic + static**	**Dynamic + FPL**
Pull-up delay (ps)	190	116
V_{OL} (mV)	0	350
Sum of logic TR sizes (μm)	43	27.3

loading effects together to the former stage's output. Worst case delay is measured including the dynamic stage delay under FO4 and 25℃ conditions and V_{OL} is also measured in the same conditions. Static logic is also simulated for the comparison. Lastly the transistor sizes are summed up to estimate the area. Results of simulations for static CMOS logic and FPL are listed in Table II. Dynamic + FPL case operates 28ps and 74ps faster than dynamic + static logic in 2-input and 3-input GG generation, respectively. But FPL dissipates much power when the logic is 0 because there is current path from V_{DD} to ground. Therefore, FPL should be applied limitedly to the critical path of the adder to keep overall power consumption low.

III. Architecture of 64bit-adder

The proposed 64-bit adder utilizes a carry-select-adder architecture which consists of conditional sum generation part and carry generation part: The two kinds of 3-bit partial sums are generated for the cases of carry 0 and 1, respectively. The optimized sum generation logic is shown in Table III. One of the conditional sums is selected according to the carry output. The carry generation part generates carries for the 3-bit partial sums, which we call the third carries (C2, C5, C8, …, C59, C61). In this section, we describe the adder architecture and the FPL is applied to this architecture for high speed carry generation.

A. *Modified tree architecture*

Kogge-Stone and Sparse tree are two famous tree architectures in 64-bit adder [7]. Kogge-Stone architecture generates carries for every bit, and it has enormous wire complexity. On the other hand, sparse-tree adder generates carries sparsely, like every third bit or forth bit, so it has much less wire complexity compared to Kogge-Stone architecture. However sparse-tree architecture concentrates many fanouts on the specific nodes like $GG_{7:0}$, $GG_{15:0}$ and $GG_{31:0}$ [8]. Especially, the number of fanout for nodes $GG_{15:0}$ and $GG_{31:0}$ are 5 and 8, respectively. Large loadings for a node in critical carry generation part degrade overall performance of the adder. Therefore, we propose Load Distribution Method (LDM) to distribute all loadings to each node fairly which diagram is shown in Fig. 5. LDM is a process that helps fanout of every node be limited to 3 and offers simple wire connections. LDM is simple and easy method but offers efficient tree architecture.

B. *Load Distribution Method*

LDM is described with our 64-bit adder as an example.

1) First, write down all carries that you want to generate. In our case, they are C2, C5, C8, …, C59, C61. Level 1 starts with C2.
2) Every node can have maximum of two branches.
3) At upper branch, generate node which is added 3*L from current node's number. L is the value of current node's level.
4) At lower branch, generate node added 3*(L+1) from current node's number.
5) Repeat 2-4 until the last carry is reached.

TABLE III. Ingredients of 3-bit conditional sum generators

Signals	When input carry is 0	When input carry is 1
C_i	G_i	P_i
C_{i+1}	$G_{i+1}+P_{i+1}\cdot G_i$	$G_{i+1}+P_{i+1}\cdot P_i$
Sum_i	$P_i\cdot /G_i$ $(=A_i \oplus B_i)$	$/P_i+G_i$ $(=/(A_i \oplus B_i))$
Sum_{i+1}	$P_{i+1}\cdot /G_{i+1} \oplus G_0$	$P_{i+1}\cdot /G_{i+1} \oplus P_0$
Sum_{i+2}	$P_{i+2}\cdot /G_{i+2} \oplus (G_{i+1}+P_{i+1}\cdot G_i)$	$P_{i+2}\cdot /G_{i+2} \oplus (G_{i+1}+P_{i+1}\cdot P_i)$

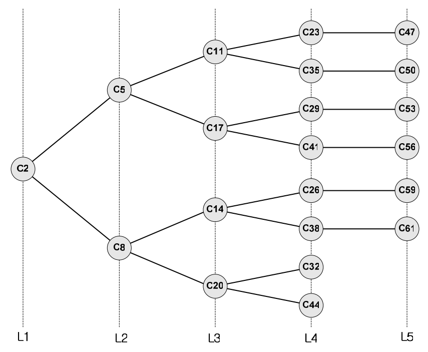

Figure 5. Concept diagram of LDM

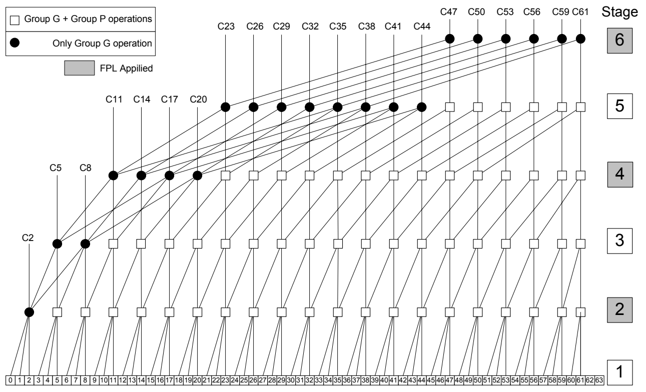

Figure 6. Modified tree architecture of 64-bit adder

Modified tree architecture of 64-bit adder using LDM is shown in Fig. 6. A dot means operation for GG generation and a square means operations for GG and GP generation. This architecture has 6-level depth including P and G generation stage.

C. *Applying FPL to the 64bit-adder*

The carry generation part of 64-bit adder is composed of 6 stages. We start the first stage with dynamic logic to make following stages behave like domino-static logic, and apply FPL to stages of static logic. Therefore, 1st, 3rd and 5th stages consist of dynamic logic and 2nd, 4th and 6th stages consist of FPL. However, in FPL stage, non-zero logic 0 voltage might cause discharge of next dynamic stage's output voltage. To prevent this and to make robust to noise, we replace 3rd and 5th to static logic. But because they still behave like dynamic logic, PUNs of these stages have only to play a role in recovering possible discharged voltage to V_{DD}. Thus we design PMOSs of these stages to have minimum widths to decrease useless loadings. After all, only first dynamic stage uses clock signal in overall 64-bit adder. We use FPL as less as possible, i.e., only in GG generation logic because FPL has power overhead compared to static logic. As a result, all carry generation part has 267 logic operations without buffer and among them 46 operations use FPL. It is only 17.23% of all logic operations. Moreover, power dissipated logic 0 evaluation will happen with the probability of 50%. Therefore, overall power overhead of applying FPL to domino-static logic is not so much as FPL itself.

IV. Simulation Results and Comparison

A 64-bit adder is implemented considering pipelining system. The adder is synchronized at positive edge of the clock signal and the previous block of the adder is synchronized at the negative edge of the clock. The outputs of the previous negative edged block have to come before set-up time of adder for correct operation. The block diagram of overall 64bit-adder and timing diagram of operations are shown in Fig. 7. In this system, we perform simulations under condition of 500MHz clock frequency, FO4 and 25 ℃ . The simulation result for worst input case is shown in Fig. 8. The last carry delay time is 258ps and final sum delay time is 372ps. Table IV summarizes the simulation results of the 64-bit adder before and after applying FPL to domino-static logic in the same architecture. FPL reduces worst case delay of the adder by 31.2% and increase power by 31%. In spite of its power overhead, performance of the 64-bit adder is significantly improved. Power-delay product and power-delay^2 product are decreased by 9.7% and 76.9%, respectively. This shows that implemented adder performs better especially in high frequencies. The comparison chart with other fast 64-bit adders in 0.18-μm technology is shown in Fig. 9. Finally, the layout of the adder is shown in Fig. 10. The adder is implemented in 0.18μm CMOS process and has area of $3.5*10^4 \mu m^2$.

V. Summary and Conclusions

We proposed a Fast Pull-up Logic for group G generation to solve a long pull-up time delay of domino-static logic adder. Using the proposed logic, we designed a fast 64-bit adder with modified tree architecture using Load Distribution Method. LDM avoids concentrating large loadings to one node and fairly distributes loadings to each node with the value of less than FO3. The implemented 64-bit adder has 372ps worst case delay and 35.82mW power consumption, which are decreased by 31.2% and increased by 31% compared to the conventional domino-static adder, respectively. This adder is one of the fastest 64bit-adders in the same process reported in the literature

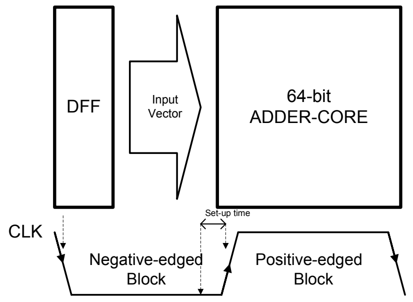

Figure 7. Overall 64bit-adder system and timing diagram

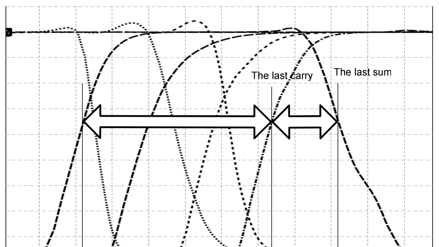

Figure 8. Simulation result of worst case delay

TABLE IV. The Simulation Results of Complete 64-bit Adders.

64-bit Adder	Domino-static	FPL applied	Comp.
Delay (ps)	540	372	-31.2%
Power (mW)	27.34	35.82	+31.0%
Power-delay product(pJ)	14.76	13.33	- 9.7%
Power-delay^2 product(10^{-9} pJs)	7.97	1.84	- 76.9%

References

[1] Hoi-Jun Yoo, "Dual-V_T self-timed CMOS logic for low subthreshold current multigigabit synchronous DRAM," IEEE Transaction on Circuits and Systems Ⅱ, Volume 45, pp.1263-1271, Sept. 1998

[2] S.-J. Lee and H.-J. Yoo, "Race logic architecture (RALA): a novel logic concept using the race scheme of input variables," IEEE JSSC, Vol.37, pp.191– 201, Feb. 2002

[3] S.-J. Lee, R. Woo and H.-J.Yoo, "480 ps 64-bit race logic adder," Digest of Tech. Papers Symposium on VLSI Circuits 2001. pp.27-28

[4] R. Woo, S.-J. Lee and H.-J. Yoo, "670ps, 64bit Dynamic Low-power Adder Design," Proceedings of ISCAS, Vol. 1, pp.28–31, May 2000

[5] Mathew, S. Anders, M. Krishnamurthy, R.K. and Borkar, S, "A 4-GHz 130-nm address generation unit with 32-bit sparse-tree adder core,"JSSC, Volume 38, pp.689 – 695, May 2003

[6] Yuke Wang, C.Pai, and Xiaoyu Song, "The Design of Hybrid Carry-Lookahead/Carry-Select Adders," IEEE Transactions on Circuits and Systems Ⅱ, Volume 49, pp.16-24, Jan.2002

[7] R. Zlatanovici, B. Nikolic, "Power-performance optimal 64-bit carry-lookahead adders," ESSCIRC 2003, pp.321 – 324

[8] Mathew, S. et al., "A 4-GHz 300-mW 64-bit Integer Excution ALU With Dual Supply Voltages in 90-nm CMOS," IEEE JSSC, Volume 40, pp.44-51, Jan. 2005

[9] Sheng Sun, et al., "409ps 4.7 FO4 64b Adder Based on Output Prediction Logic in 0.18um CMOS" Proceedings of IEEE Computer Society Annual Symposium on VLSI pp.52 – 58, May 2005.

[10] A. Neve, H.Schettler, T.Lugwig, D.Flandre, "Power-delay product minimization in high-performance 64-bit carry-select adders," Trans. VLSI , March 2004, pp.235 – 244.

[11] D. Stasiak, F. Mounes-Toussi, S.N.Storino, "A 440-ps 64-bit adder in 1.5-V 0.18um partially depleted SOI technology,"JSSC, Oct.2001,

[12] J.Park, H.Ngo, J.Silberman, S.Dhong, "470-ps 64-bit parallel binary adder," Digest of Symposium on VLSI Cir., pp. 192-193, June 2000

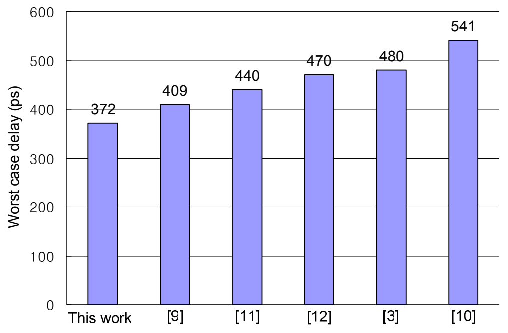

Figure 9. Comparison chart with other fast 64-bit adders

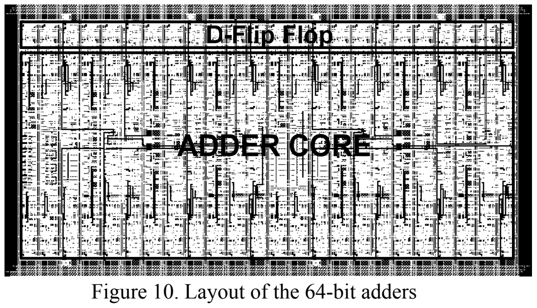

Figure 10. Layout of the 64-bit adders