High Speed 1-bit Bypass Adder Design for Low Precision Additions

Jong-Suk Lee and Dong Sam Ha
VTVT (Virginia Tech VLSI for Telecommunications) Lab
Department of Electrical and Computer Engineering
Virginia Tech, Blacksburg, VA 24061
E-mail: {watsup, ha}@vt.edu

Abstract—**In this paper, we propose a high speed adder which is adopted for our reconfigurable architecture called FleXilicon. To support sub-word parallelism, the FleXilicon architecture adopts 8-bit processing units as the atomic operation. Hence, high speed 8-bit adders are a key building block necessary for high performance. The proposed adder intends to speed up 8-bit adder operations. It is based on a conventional bypass adder scheme, but bypasses 2 bits on every adder bit stage rather than bypassing 4 bits on every four bit stages for conventional bypass adders. The proposed adder enables high speed operation for the FleXilicon and maximizes sub-word parallelism. The proposed adder is implemented with full custom design in CMOS 65 nm process. Simulation results show that the proposed adder is two times faster than existing adders for 8 bit additions.**

I. INTRODUCTION

We developed a reconfigurable architecture called FleXilicon for multimedia and wireless communication applications [1]. It is a coarse grained architecture, which provides massive parallel processing of loops. The FleXilicon can support flexible word-length operations, which can maximize the sub-word parallelism (SWP). The SWP is a method to increase the parallelism by partitioning a datapath into sub-word, so that multiple sub-word data can be processed concurrently [2]. Therefore, the SWP can be used effectively for parallel processing of the various precision data. To support SWP efficiently, unlike conventional reconfigurable architectures with 16 or 32 bits as the size for atomic operations, 8-bit processing units are used for the atomic operation for the FleXilicon architecture. Using up to 32 eight-bit atomic processing units, the FleXilicon offers highly flexible SWP. Hence, a high speed 8-bit adder is the essential building block for high performance of FleXilicon. In addition, low precision adders are also an important building block for processors and ASICs, since various types of additions (such as execution unit, address generation unit, branch predictor, and datapath) often require low precision additions. It should be noted that VLIW architectures, SIMD architectures and multimedia extension hardware for microprocessors also support multiple low precision operations [3]-[6].

Existing high speed adders focus on wide bit-width additions such as CLA (Carry Look-ahead Adder) [7], bypass adder [7], BK (Breton Kung) [8], PPrefix (Parallel Prefix) [9], whose speed improvement is rather insignificant for low precision additions. Further, some existing adders such as CLA [7] and bypass adders [7] incur substantial additional logic for the speed up, but the additional logic slows down low precision additions. Previous tree adders such as BK [8] and PPrefix [9] adders minimize the depth of logic, but it is ineffective in speeding up low precision additions.

While existing adders focus on high performance for high precision additions, we propose a high speed adder for low precision additions such as 8-bit and 16-bit additions. The proposed adder supports high speed operation for SWP, which is a basic processing unit for the FleXilicon architecture. This paper is organized as follows. Section 2 describes design of the proposed adder and analyzes the operation. Section 3 presents simulation results and compares the performance of the proposed adder with conventional adders. Section 4 draws a conclusion.

II. PROPOSED ADDER

In this paper, we investigated speed optimization for a low precision adder such as 8-bit adder and a 16-bit adder. To achieve high speed for a low precision adder, we propose a 2-bit bypass scheme one every bit stage instead of a typical 4-bit bypass scheme on every 4 bit stages. Figure 1 shows our 8-bit adder based on 2-bit bypass. It is constructed with a HA (Half Adder) and seven FA (Full Adder) cells. A FA cell consists of three sub-blocks, *a carry control signal block*, *a carry path block,* and *a summation block*.

The carry control signal block, which is the top block in the figure, generates signals to control the carry path circuit. It generates four control signals, Kill (K), Generation (GX), Propagation (P), and Bypass (B). A carry path block, which is the middle block, consists of a chain of three sub-circuits, a transmission gate for carry propagation, a carry kill/generation circuit, and a bypass circuit for a bypass. The summation block, i.e., the bottom block, is simply an Exclusive-OR gate, which generates the sum value using signal P and the carry.

The transmission gate of a carry path block propagates (blocks) the previous carry under P=1 (P=0). The carry kill/generation circuit drives the carry signal path to 1 if GX

1-4244-0921-7/07 $25.00 © 2007 IEEE.

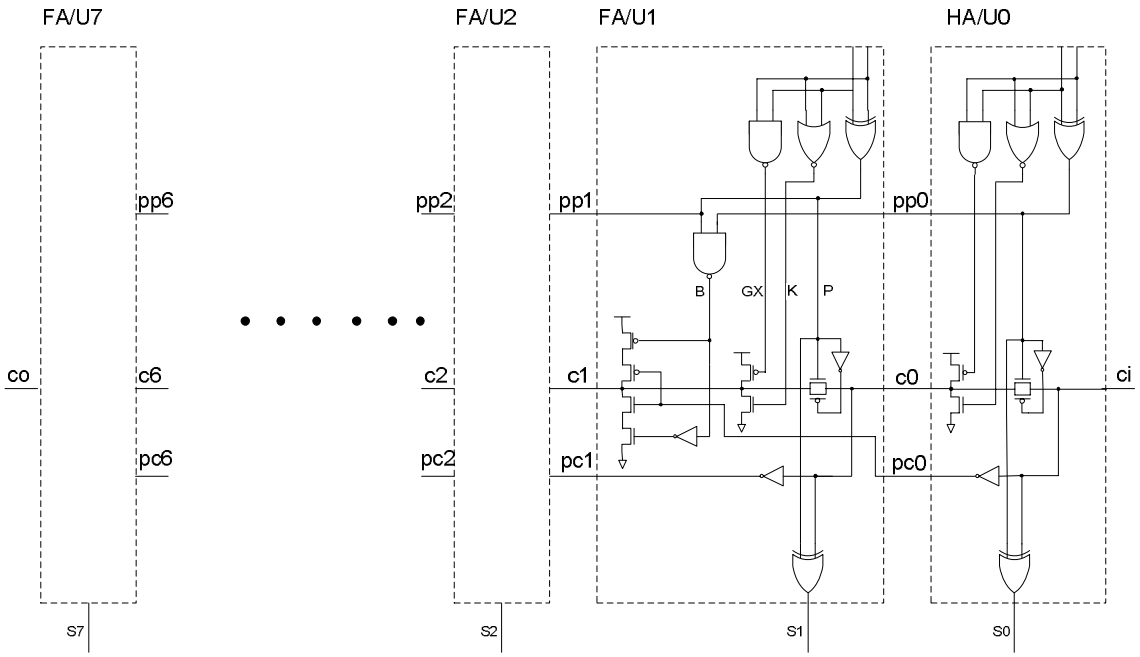
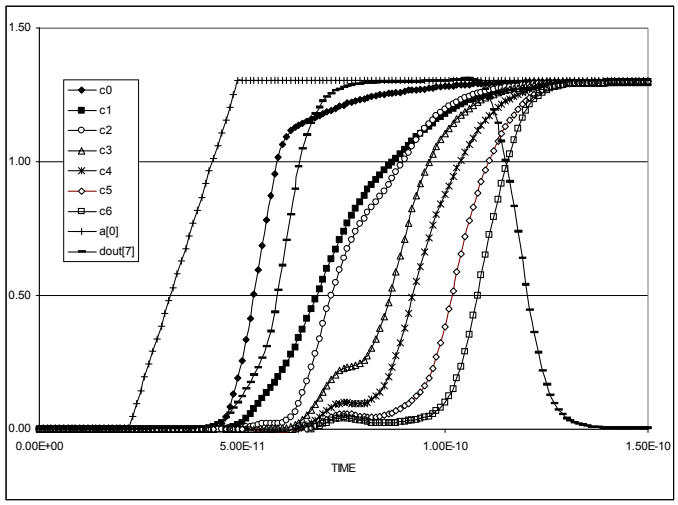


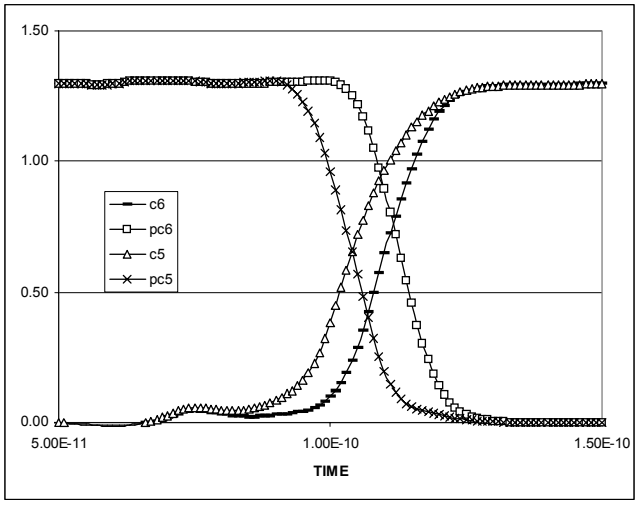
Figure 1: Proposed 8-bit adder

is 0 (which is "Generation" is true) and 0 if K=1 (which is "Kill" is true.) independent of the carry propagated from the transmission gate. The bypass circuit bypasses the carry signal for two bit stages. Signal B becomes 0 if both pp*i* and pp*i-1* are 1. Under B=0, the carry c*i-2* bypasses the next two stages, *i-1* and *i*, and drives the carry signal c*i* directly.

The proposed adder is similar to the Manchester carry-chain adder [7], besides the proposed adder has a bypass path. Our bypass adder has two key differences from conventional bypass adders. First, our adder bypasses two bits, but the bypass can occur on every bit stage. Note that the bypass for a traditional n-bit bypass adder occurs on every n-bit stages, for example, ever 4 stages for a 4-bit bypass adder. The other difference is that a carry path and a bypass path share the same path in the carry path block for the proposed adder, while a carry path is isolated from a bypass path for conventional bypass adders. Sharing the same path results in both the pc*i-1* and the c*i-1* signals from the stage *i-1* drive the same carry signal node c*i* of the stage *i*. The proposed scheme works, as the value driven by an inverter through the



(a) Waveforms of carry signals



(b) Waveforms of carrier and carry bypass signals

Figure 2: Path delays of the proposed adder

signal pc*i-1* arrives first and overrides the signal value driven by a transmission gate through the c*i-1*.

Figure 2 shows the waveforms of SPICE simulation for input *A = 00000001* and input *B = 11111111* with carry in c*i*=0. For those particular inputs, a carry is generated at the stage 0 and propagate all the way to the stage 7. The carry signal c*0* is generated by the kill/generation circuit of the stage 0 and drives the carry c*1* through the carry path. The bypath signal pc*0* does not contribute the c*1* value, since the driving inverter (with input pc*0* and output c*1*) is disabled. As the result, the transition time of c*1* is greater than that of the carry signal c*0*. Subsequent carry signals are bypassed by two bits. The carry signal c*2* is bypassed from c*0* through pc*1*, the carry signal c*3* from c*1* through pc*2,* and so forth. At the same time, those carry signals are also driven by the corresponding previous carry signals through the transmission gates of the carry path. Figure 2 (b) shows how the proposed bypass scheme achieves the speed up. The carry signal c*6* driven by the signal pc*5* starts to change even earlier than signal pc*6*, which is the inverted carrier signal c*5*. This means that carry propagation delay from the carrier signal c*5* to the carrier signal c*6* is less than one inverter delay.

The worst case delay for the proposed n-bit adder can be estimated roughly using Equation (1).

$$Delay = (n-2)\times d' + d'' \quad (1)$$

In the equation, n is the bit width of an n-bit adder, d' is the carry propagation delay of one stage except the first two LSB stages, and d'' the carry propagation delay of the first two LSB stages. As shown in waveform in Figure 2 (a), the delays of the first two stages are much larger than the rest of stages. Our simulation result with a 65 nm CMOS SPICE model shows that the carry propagation delay of one stage d' is 9.2 ps and the delay of the first two stages d'' is 28.9 ps or 14.5 ps for each sage. We also noticed that the FO4 inverter delay for the same technology is 11.5 ps. Noting the delay of one stage d' is 9.2 ps, the proposed adder operates faster than an inverter chain with same number of stages except the first two LSB stages.

III. Performance Comparison

In this section, we present comparison of performance for the proposed adder and existing adders such that BK (Breton Kung) [8], PPrefix (Parallel Prefix) [9], CLA (Carry Look-ahead Adder) [7], and RPL (ripple carry adder). BK and PPrefix adder are tree adders, which minimize the depth of the carry path by employing parallel computation of the carry. The CLA uses look-ahead logics for carry generation which saves the propagation delay. We synthesized the BK, PPrefix and CLA using Synopsys DesignWare library [10] and implemented a RPL at a schematic level using a standard cell library. Critical path delays of synthesized adders were obtained through static timing analysis using Synopsys Primetime [11]. The delay of the proposed adder was obtained through SPICE simulation. To calibrate static timing analysis results with SPICE simulation results, the delay of the ripple carry adder was simulated in both methods and compared the results each other to obtain a calibration factor.

Figure 3 shows critical path delays of various types of 8 bit adders in a CMOS 65 nm technology under the supply voltage of 1.3 V and the temperature of 100 C. The delay of the proposed adder is only 84 ps and the smallest among the five adders compared, while the delays of the other adders lie in the range of 167 ps to 188 ps. The speedup of the proposed adder over other types is about two times, which contributes the high speed operation for the FleXilicon.

To construct a larger adder using the proposed bypass adder, we can simply concatenate multiple FA (Full Adder)

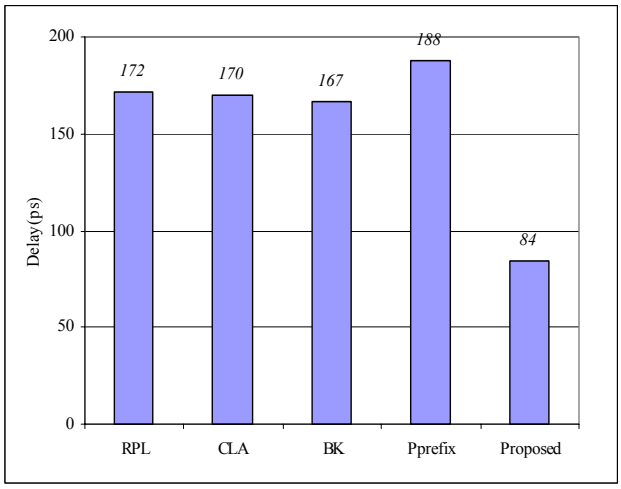


Figure 3 Critical path delays of 8-bit adders.

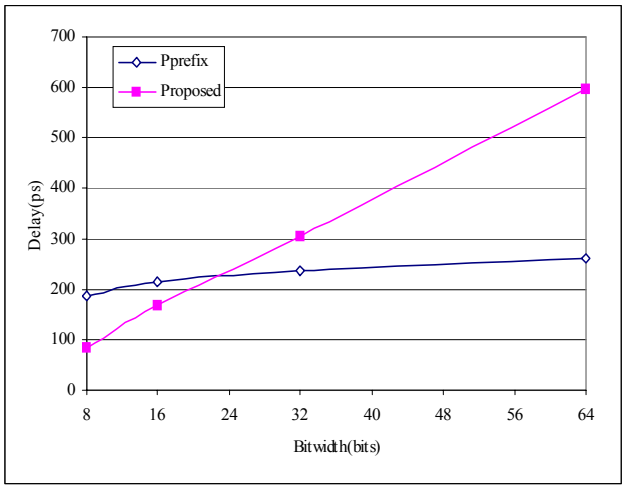


Figure 4: Critical path delay of multibit adders.

units. Figure 4 shows simulation results of larger adders, up to 64 bits, constructed of the proposed FA units, and the results are compared with that of a PPrefix adder. As shown in the figure, the proposed adder achieves higher performance than the PPrefix adder for 8-bit and 16-bit adders, but the proposed adder performs worse beyond the size. As shown in the graph, the delay of the proposed adder increases linearly with the increase of the bit-width, which confirms the validity of Equation (1). It should be noted that the depth of the carry path of a PPrefix adder increases in a logarithm scale.

IV. CONCLUSION

In this paper, we proposed a high speed adder which is adopted to support sub-word parallelism for our FleXilicon architecture. The proposed adder intends to speed up 8-bit adder operations. It is based on a conventional bypass adder scheme, but bypasses 2 bits on every adder bit stage rather than bypassing 4 bits on every four bit stages for conventional bypass adders. The proposed adder is implemented with full custom design in CMOS 65 nm process. Simulation results show that the proposed adder is two times faster than existing adders for 8 bit additions and performs better than a PPrefix adder up to 16 bits. It is also possible for a hybrid implementation of our adders with other types of adders to improve the speed for higher precision additions.

V. REFERENCES

[1] J-S. Lee, D.S. Ha, "FleXilicon: a Reconfigurable Architecture for Multimedia and Wireless Communications", *International Symp. Circuits and Systems*, May. 2006, pp. 4375-4378.

[2] J. Fridman, "Sub-word Parallelism in Digital Signal Processing", *IEEE Signal Processing Magazine*, Mar. 2000.

[3] S. Agarwala, et al., "A 600-MHz VLIW DSP", *IEEE Journal of Solid-state Circuits*, Vol 37, No. 11, November 2002.

[4] D.A. Draper and et. al., "An X86 microprocessor with multimedia extensions", *IEEE International Solid-State Circuits Conference*, Feb. 1997, pp. 172-173.

[5] S. Oberman, G. Favor, and F. Weber, "AMD 3D Now! Technology: architecture and implementations," *IEEE Micro*, vol. 19 Apr. 1999, pp. 37-48.

[6] R.B. Lee, A. M. Fiskiran, and A. Bubshait, "Multimedia instructions in IA-64", *Proc. Int. Conf. Multimedia Expo*, Aug. 2001.

[7] N.H.E Weste, K. Eshraghan, *Principles of CMOS VLSI Design*, Addison Wesley, 1993.

[8] R.P. Brent and H.T. Kung, "A Regular Layout for Parallel Adders", *IEEE Trans. Computers*, Vol. 31, Mar. 1982, pp. 260-264.

[9] G. Dimitrakopoulos, D. Nikolos, "High-Speed Parallel-Prefix VLSI Ling Addres", *IEEE Trans. Computers*, Vol. 54, No. 2, Feb. 2005, pp 225-231.

[10] DesignWare library User's Guide, Available: http://www.synop-sys.com/

[11] PrimeTime User' Guide, Available: http://www.synopsys.com/