Design and Implementation for Fast Adder

Prateek Sahu[1] Yuwei Liu[1] Chiaju Chen[1]

***Abstract*— In this paper, we proposed a new design for an adder circuit, which significantly decrease the delay time for increment and decrement operations.**

I. Introduction

Adders are typically a single cycle operation as opposed to multiplications and divisions. One of the most common use of adders in processor design is for increment and decrement operations which are required to manage a variety of data structures like stack, FIFOs, queues etc., other than normal ALU operations. We would like to reduce the critical path delay for an increment and decrement operation because these are common operations and hence help in reduction of critical path delay.

II. Probelem Description

Most stack operations in computer systems are increment and decrement operations by a fixed amount. Pointers in FIFO operations are also a unit increment operation. In a 32-bit system, all these operations are implemented using a 32-bit Adder. These operations take 32-bit adder delay. But these adders can be specially treated to be faster and have less complexity. We proposed one such method to reduce the delay and complexity of an n-bit incrementor and decrementor.

A. Intuition

- An n-bit incrementor can be represented as $A + B$, where A is the value being incremented and B is n-bit 1. This can be also be represented as $A + 0$ and a Cin of 1. With B being 0, the full adder circuit significantly reduces in complexity and delay.
- Take a standard carry lookahead adder with 4-bit block size as example:

$$P = p_0 \cdot p_1 \cdot p_2 \cdot p_3 \tag{1}$$

$$G = g_3 + p_3 \cdot g_2 \cdot g_1 + p_3 \cdot g_2 \cdot g_1 + p_3 \cdot g_2 \cdot g_1 \cdot g_0 \tag{2}$$

$$p_i = a_i \oplus b_i, g_i = a_i \cdot b_i \tag{3}$$

When given B = 0, it yields the result that,

$$g_i = 0, p_i = a_i, G = 0, P = AND(a_i) \tag{4}$$

*This work was not supported by any organization

[1] Electrical and Computer Engineering, Cockrell School of Engineering, University of Texas at Austin

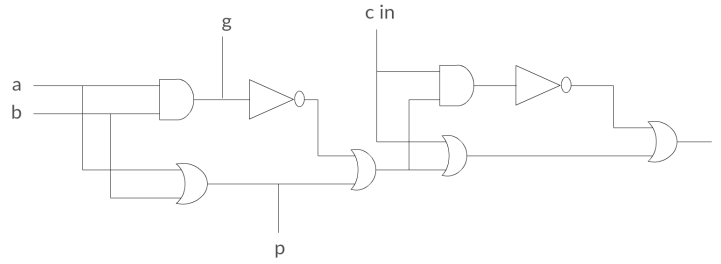

Fig. 1. CLA Full Adder Unit

This simplification makes the specialized circuit much simpler and has a much faster implementation. Similarly, a decrementor can be represented as,

$$S = A + \underbrace{111...1}_{n} \tag{5}$$

where B is the second term in equation 5. This also reduces to,

$$P = 1, G = OR(a_i) \tag{6}$$

and it can be implemented as a fast adder.

B. Circuit Derivation

Fig 1 shows a standard Full Adder circuit used in most Carry Look Ahead Adder designs, the propagate p and generate g signals are directly dependent on just the a and b inputs of the adder and not the carry-in cin signals. These p and g signals are then further propagated to subsequent levels of lookahead logic to calculate the carry-out signals faster, hence making this adder design significantly faster than Ripple Carry Adder.

- Carry Lookahead Adder logic
  The CLA logic helps calculate the carry outs of each bit faster by the following logic equations.
  Given 4 bits carry lookahead adder, we can get:

$$c_4 = g_3 + p_3 \cdot g_2 + p_3 \cdot p_2 \cdot g_1 + p_3 \cdot p_2 \cdot p_1 \cdot g_0 + p_3 \cdot p_2 \cdot p_1 \cdot p_0 \cdot c_0 \tag{7}$$

$$c_3 = g_2 + p_2 \cdot g_1 + p_2 \cdot g_1 \cdot g_0 + p_2 \cdot p_1 \cdot p_0 \cdot c_0 \tag{8}$$

$$c_2 = g_1 + p_1 \cdot g_0 + p_1 \cdot p_0 \cdot c_0 \tag{9}$$

$$c_1 = g_0 + p_0 \cdot c_0 \tag{10}$$

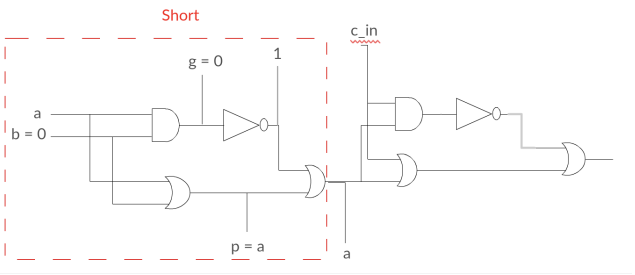

Fig. 2. CLA Modified Full Adder Circuit – Increment

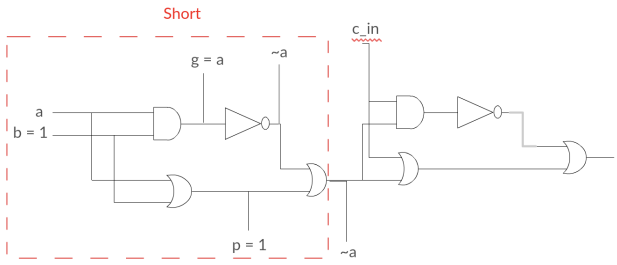

Fig. 3. CLA Modified Full Adder Circuit – Decrement

As we see, we use 2input, 3input, 4input AND and OR gates for these equations which are standard gates and hence use 4 as the block size for CLA logic.

- Increment Operation
  For the increment operation, set $B = 0$, we can get $g_i = 0$ and $p = a_i$ as shown in Fig. 2. In the above equations 7, 8, 9 and 10, if we substitute equation 4, we get the following set of equations 11, 12, 13, 14, which make up the increment adder logic and simplifies the c_i term.

$$c_4 = p_3 \cdot p_2 \cdot p_1 \cdot p_0 \cdot c_0 = a_3 \cdot a_2 \cdot a_1 \cdot a_0 \tag{11}$$

$$c_3 = p_2 \cdot p_1 \cdot p_0 \cdot c_0 = a_2 \cdot a_1 \cdot a_0 \tag{12}$$

$$c_2 = p_1 \cdot p_0 \cdot c_0 = a_1 \cdot a_0 \tag{13}$$

$$c_1 = p_0 \cdot c_0 = a_0 \tag{14}$$

- Decrement Operation
  Similarly for the decrement operation, set $B = 1$, we can get $g_i = a_i$ and $p_i = 1$ as shown in Fig 3. In the above equations 7, 8, 9 and 10, if we substitute 6, we get the following set of equations 15, 16, 17 and 18, which make up the increment adder logic and simplifies the c_i term.

$$c_4 = p_3 + p_2 + p_1 + p_0 = a_3 + a_2 + a_1 + a_0 \tag{15}$$

$$c_3 = p_3 + p_2 + p_1 + p_0 = a_2 + a_1 + a_0 \tag{16}$$

$$c_2 = p_1 + p_0 = a_1 + a_0 \tag{17}$$

$$c_1 = p_0 = a_0 \tag{18}$$

Above derivation can fast bypass the calculation of carries and deduce the delay time.

C. Scale up

We showed the basic unit for 4 bits, and we now expanded the design to 32 bits. From CLA lookahead logic, we get the expressions of 4-bit block P, G and C as equation 19, 20 and 21. By utilizing equation 4, we can reduce them to equation 22, 23 and 24. Therefore, the final 32-bit modified CLA with 4-bit modified unit and lookahead logic is shown as figure 4.

$$g_{k:k+3} = g_{k+3} + p_{k+3} \cdot g_{k+2} + p_{k+3} \cdot p_{k+2} \cdot g_{k+1} + p_{k+3} \cdot p_{k+2} \cdot p_{k+1} \cdot g_k \tag{19}$$

$$p_{k:k+3} = p_{k+3} \cdot p_{k+2} \cdot p_{k+1} \cdot p_k \tag{20}$$

$$c_{k:k+4} = g_{k:k+3} + p_{k:k+3} \cdot c_k \tag{21}$$

$$g_{k:k+3} = 0 \tag{22}$$

$$p_{k:k+3} = a_{k+3} \cdot a_{k+2} \cdot a_{k+1} \cdot a_k \tag{23}$$

$$c_{k:k+4} = p_{k:k+3} \cdot c_k \tag{24}$$

III. Experiment Setup

The 8-bit, 16-bit, 32-bit fast adders were implemented in Structural Verilog, and simulated using VCS [1] tool to verify the functional correctness. The synthesis results from Design Vision [2] tool reported the design specifications like total layout area, power and critical path for the design. All the results have been reported in tables I and II. Both tools are from Synopsys and available on UT ECE LRC. In terms of metrics of synthesis results, we chose timing, area and power, to compare that of our implementation of traditional CLA.

For Design Vision, it provides 45nm standard cell library with notations of timing, area and power. We used these gate-level cells as basic modules to implement our design. All designs use 2 input gates only. Any 3 input and 4 input gates used for carry calculations are designed using 2 input gates in structural verilog.

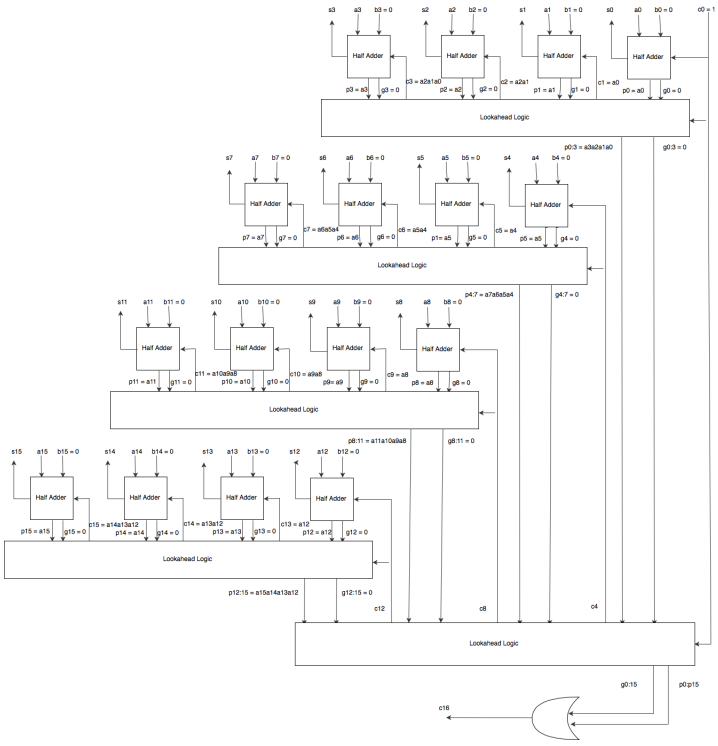

Fig. 4. 32bits Modified CLA

IV. Experiment Results

Timing and area are measured in Design Vision specific units. From table I and II, we can clearly see a 50% decrease in the critical path timing for each of the increment and decrement adder compared against traditional CLA adders. This is an obvious deduction since the Full Adder circuit for a CLA is reduced to a Half Adder for each of the increment and decrement designs. Because of this, the complexity and hence the area also reduces significantly.

We still see increase in all the three parameters when we move from 8-bit to 16-bit to 32-bit designs in the same ratio as a traditional CLA does since our modified designs implement CLA logic at the heart of it.

Due to the major reduction in the number of gates, the the switching power and the leakage power both drop significantly giving us an clear power benefit.

TABLE I

Results for fast adder

	Timing	Area	Power(mW)
increment 8bits	0.3	103.25	1.4546e-02
increment 16bits	0.39	202.73	2.5335e-02
increment 32bits	0.61	452.41	4.7819e-02
decrement 8bits	0.3	97.15	1.4921e-02
decrement 16bits	0.4	192.88	2.6223e-02
decrement 32bits	0.63	431.29	4.9007e-02

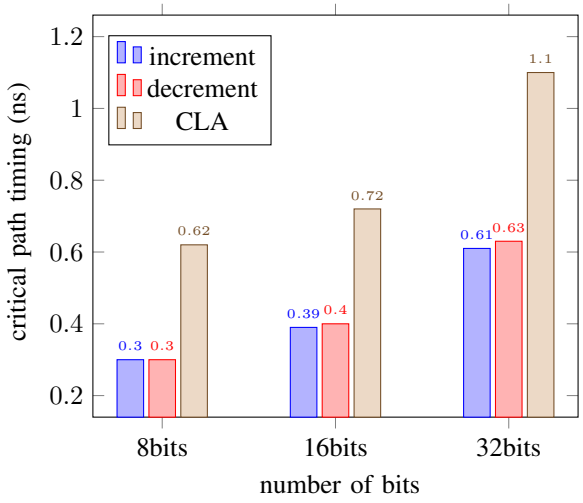

Fig. 5. Comparison of timing

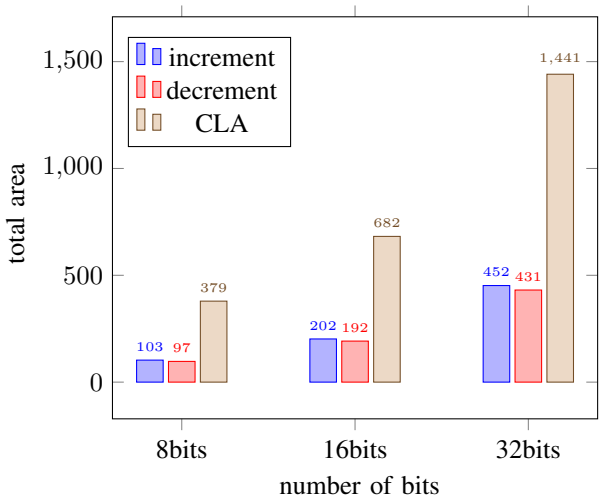

Fig. 6. Comparison of area

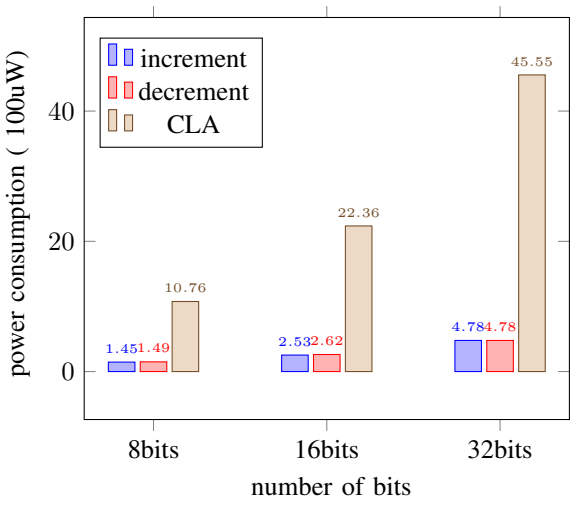

Fig. 7. Comparision of power

TABLE II

RESULTS FOR TRADITIONAL CLA

	Timing	Area	Power(mW)
CLA 8bits	0.62	379.19	0.1076
CLA 16bits	0.75	682.83	0.2236
CLA 32bits	1.1	1441.69	0.4555

V. CONCLUSIONS

With empirical data backing our initial assumptions, we can strongly claim that, such specialized data-dependent adder circuits are a clear advantage over traditional general purpose adders. Such designs use less power and area and can provide great advantage to processor design community if used appropriately. Apart from CLA, such reduction techniques can also be extended to all families of adders derived from CLA based designs including Parallel Prefix adders like Kogge-Stone [5] and Brent-Kung [6].

We plan to implement designs for such adders and do a comparative study for the same in the future and deliberate on the advantage of specialized circuit design over traditional general purpose designs.

REFERENCES

[1] VCS, Synopsys, https://www.synopsys.com/verification/simulation/vcs.html

[2] Design Vision, Synopsys, https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test/design-compiler-graphical.html

[3] Doran, Robert W. "Variants of an improved carry look-ahead adder." IEEE Transactions on Computers 37.9 (1988): 1110-1113.

[4] Ohkubo, Norio, Makoto Suzuki, and Katsuro Sasaki. "Carry look ahead circuit." U.S. Patent No. 5,424,972. 13 Jun. 1995.

[5] Kogge, Peter M., and Harold S. Stone. "A parallel algorithm for the efficient solution of a general class of recurrence equations." IEEE transactions on computers 100.8 (1973): 786-793.

[6] Brent, Richard P., and H-T_ Kung. "A regular layout for parallel adders." IEEE transactions on Computers 3 (1982): 260-264.