Super Pipelined Digit Serial Adders for Multimedia and e-Security

Adnan Abdul-Aziz Gutub, Mohammad K. Ibrahim, and Muhammad Amer Araman

Department of Computer Engineering
King Fahd University for Petroleum & Minerals, Dhahran 31261, Saudi Arabia
gutub@kfupm.edu.sa

Abstract

This paper presents super-pipelined models of conventional adders that use digit serial addition. We pipeline three adders: ripple carry, carry select, and carry lookahead showing the pipelining effect in their speed and area. An improvement to the pipelined carry lookahead adder is proposed showing interesting results.

1. Introduction

Digit serial adders are frequently used in computer systems that need to deal with operands of large sizes such as multimedia signal processing and e-security applications [1,2,9,11]. To add two large operands, the operands are divided into a number of small digits that can be summed sequentially using small or medium sized adders. Every digit summation needs the carry resulted from the less significant digits [4]. This carry feedback makes if very difficult to pipeline the addition of digits within the adders, which means that the addition of two digits must completely finish before the addition of the more significant digits can start. Figure 1 shows the architecture of a conventional digit serial adder.

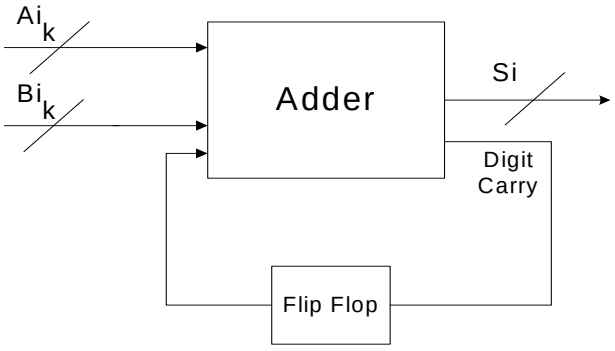

Figure 1: normal digit serial adder

The carry feedback can be eliminated if two adders are used in sequence as in Figure 2. In this design the carry is fed forward to the second adder and added to the next digit output as detailed in [7]. We can increase the level of pipelining by introducing stages within each adder block to increase the clock rate and get a higher throughput.

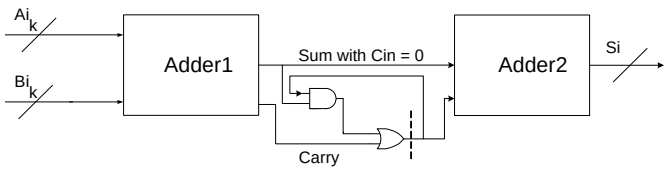

Figure 2: Pipelined Digit Serial Adder

This paper discusses the possibility of using super-pipelined versions of commonly used adders (ripple carry, carry select, and carry lookahead) for digit serial addition. A new design using a modified carry lookahead adder is presented and compared with conventional pipelined adders.

2. Ripple Carry Adder

The pipelined ripple carry adder (RCA) consists of *k* stages where *k* is the digit size as clarified in [4]. The time for each pipeline stage is dictated by the delay of one full adder (FA), which is assumed as 3 gates. Note that the time and area would duplicate if the same RCA were used in the second adder of Figure 1.

This adder is good for only small digit sizes because the area increases dramatically as the digit size increases [1,5,6,8,9,10]. Table 1 lists the time and area assumptions to be used throughout this

Device	Area	Delay
FA (Full Adder)	5 gates	3 gates
Latch	5 gates	2 gates

Table 1: Components delay and area assumptions

No. of stages	Stage time	Latency	Area	Time non-piplined for 40 digits	Time piplined for 40 digits	Speed Up
8	5	40	466	480	235	2.04
4	8	32	170	480	344	1.4

Table 2: tradeoffs for 4-bit RCA digit serial adder

No. of stages	Stage time	Latency	Area	Time non-piplined for 10 digits	Time piplined for 10 digits	Speed Up
32	5	160	4090	480	205	2.34
8	14	112	1090	480	238	2.02
4	26	104	590	480	338	1.42

Table 3: tradeoffs for 16-bit RCA digit serial adder

paper using the gate count as its unit. Gate counts are used to be completely independent to the technology improvement. Table 2 and Table 3 present various tradeoffs for 4-bit adder and 16-bit adder respectively.

3. Carry Select Adder

This adder uses to RCA adders that work in parallel [3]. The first RCA find the sum with carry in equal zero, while the second RCA finds the sum with carry in equals 1. The carry of the previous digit is used to select the correct result. This will replace the second adder of Figure 2 with a multiplexer as shown in Figure 3.

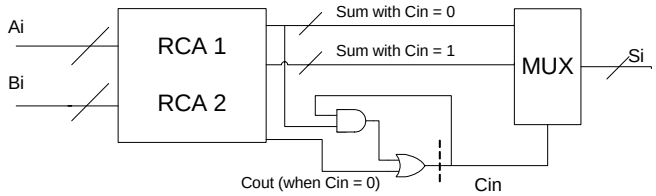

Figure 3: Pipelined Carry Select Adder

The carry to next digit is one in a case of two:

1. Carry is generated (*Cout = 1* when *Cin = 0*)
2. Carry is propagated from previous carry. This is checked by ANDing the Sum with *Cin = 0*, which is XOR of the digits to previous carry.

Although Figure 3 design adds the Mux area to the adder of Figure 2, there is a considerable decrease in the latency. Table 4 and Table 5 present the latency, area and speedup of pipelined Carry Select Adder (CSA) with different numbers of stages for addition of 160-bit operands where 8 pipelined 4-bit RCA adders are operated in parallel (2 stages). One stage is added to select the correct sum and carry output from each RCA. The second stage chooses the correct result through the Mux. Notice that we have two vectors of sums and carries, one for *carry-in=0* and one for *carry-in=1*. The area and speed change of the pipelined CSA are listed in Table 5.

We can reduce the latency even more if the adder is subdivided into a number of carry select blocks. All the summing is done in parallel and the carry ripples through the blocks to select the correct result from each block as in Figure 6.

4. Carry Lookahead Adder

The carry lookahead adder (CLA) is detailed in [11]. It consists of an array of partial full adders to calculate G_i and P_i for all i from *0* to *k-1* in parallel such that

No. of stages	Stage time	Latency	Area	Time non-piplined	Time piplined	Speed Up
4	5	20	474	480	215	2.232558
2	8	16	178	480	328	1.463415

Table 4: tradeoffs for 4-bit digit serial Carry Select Adder

No. of stages	Stage time	Latency	Area	Time non-piplined	Time piplined	Speed Up
16	5	80	4122	480	125	3.84
4	14	56	1122	480	182	2.637363
2	26	52	622	480	286	1.678322

Table 5: tradeoffs for16-bit digit serial Carry Select Adder

$G_i = A_i \cdot B_i$ Generate a carry

$P_i = A_i \oplus B_i$ Propagate pervious carry

All *G*s and *P*s are fed into a CLA logic as shown in Figure 4. This CLA logic calculates all the carries in parallel, which are then fed back to the partial adders to find the sum bits from the following equation assuming the first carry-in is called C_0

$S_i = C_i \oplus P_i$

In order to pipeline the CLA, we redraw the adder as a three-stage pipeline [4]. The first stage finds all *P*s and *G*s, the second stage calculates the carries and the third stage finds the sum by XORing *P*s and *C*s. This design is useful for small number of bits [5,6]. However, when the size of digit is increased, the carry lookahead logic becomes impractical because it will have logic gates with a large number of inputs [1,2,5].

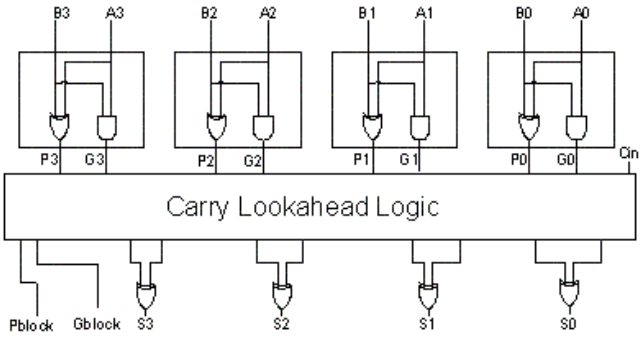

Figure 4: Pipelined 4 bit Carry Look Ahead Adder

The CLA logic also finds *G*block that indicates whether the block has generated a carry, and *P*block that indicates whether the block propagates the carry from its previous digit to its following digit.

For a 4-bit CLA:

$P_{Block} = P_3 \cdot P_2 \cdot P_1 \cdot P_0$

$G_{Block} = G_3 + G_2 \cdot P_3 + G_1 \cdot P_2 \cdot P_3 + G_0 \cdot P_3 \cdot P_2 \cdot P_1$

These bits are useful to propagate carries between digits in our digit serial adder as shown in Figure 5.

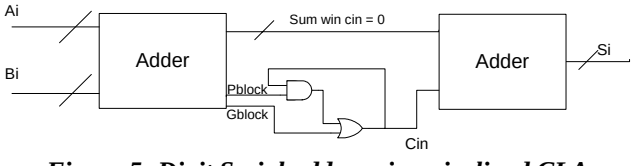

Figure 5: Digit Serial udder using pipelined CLA

For large digit sizes, a number of smaller blocks are used instead of building a CLA for the whole digit. The pipeline for two level blocks is as follows:

S1: find all Ps and Gs
S2: find Pblock and Gblock for all blcoks
S3: find Cin for all CLAs
S4: find bit level Cs
S5: Si = Ci xor Pi

This pipeline can be extended for any number of block levels. Note that the logic for finding *P*block and *G*block is separated from the logic that finds the carries to have a one directional movement of data.

5. Modified Carry Lookahead Adder

To eliminate the need for the second adder in Figure 5, the CLA logic is modified to find two values of sums, i.e. sum with *Cin=0* and sum with *Cin=1*. Off course, this will add in the area and complexity of the hardware. However, it turned out that there is no need to duplicate the complete area for carry-in when applied to the gates. All OR gates connected to carry-in which assumed *Cin=1* will replace the gate by simple connection to logic one. Similarly, all AND gates with carry-in as zero (*Cin=0*) will be wired to logic zero. These carry-in effects, interestingly, resulted in an area similar to the non-pipelined CLA. This modified carry generation level logic is shown in Figure 6.

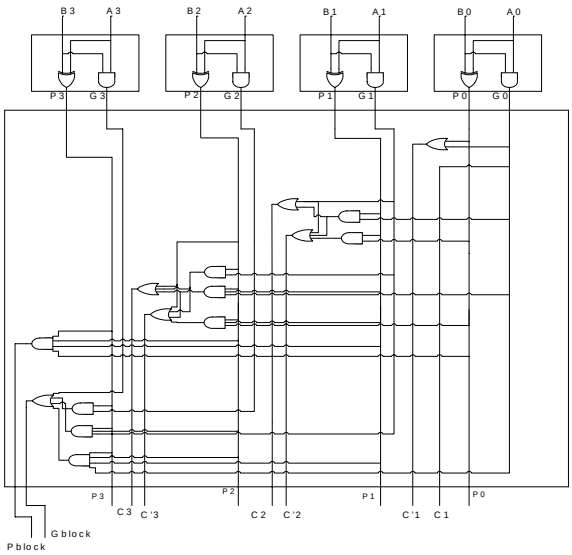

Figure 6: 4-bit Modified CLA

No. of stages	Stage time	Latency	Area	Time non-piplined	Time piplined	Speed Up
5	5	25	1424	480	70	6.857143
3	8	24	864	480	96	5

Table 6: ***Latency, Area and Speedup for 16-bit blocked pipelined Carry Select Adder***

The area, delay, and speedup of different models of this modified CLA are given in Table 6 for the same number of bits. The results show that our improved pipelining leads to interesting speedup with no significant increase in area.

6. Conclusions

It is possible to apply super-pipelining to most of the common adder design. To super-pipeline a digit adder, the carry is fed forward to another adder rather than back to the same adder. Digit serial adders were designed using Ripple Carry Adder, Carry Select Adder, and Carry Lookahead Adder and Modified Carry Lookahead adder. It is found that our modified Carry Lookahead Adder enjoys very interesting speedup with no considerable expansion in area. However, it has the limitations of Carry Lookahead logic such as large gate fan-in and non-regularity.

Acknowledgment

The authors are grateful to King Fahd University of Petroleum and Minerals (KFUPM) for supporting this work.

References

[1] Crawley, D.G., and Amaratunga, G.A.J. : '8x8 Bit Pipelined Dadda Multiplier in CMOS', IEE Proceedings - Circuits, Devices and Systems, Dec. 1988, 135, (6), pp. 231-240

[2] Crawley, D.G., and Amaratunga, G.A.J. : 'A Standard Cell Automated Layout for a CMOS 50MHz 8x8 Bit Pipelined Parallel Dadda Multiplier', IEEE International Symposium on Circuits and Systems ISCAS'88, June 1988, 1, pp. 977 – 980

[3] Youngjoon Kim, Ki-Hyuk Sung, and Lee-Sup Kim : 'A 1.67 GHz 32-bit Pipelined Carry-Select Adder Using the Complementary Scheme', IEEE International Symposium on Circuits and Systems ISCAS'2002, 1, May 2002, pp. 461-464

[4] Dadda, L., and Piuri, V.; 'Pipelined Adders', IEEE Trans. Computers, March 1996, 45, (3), pp. 348 – 356

[5] Lopez, J.F., Reina, R., Hernandez, L., Tobajas, F., de Armas, V., Sarmiento, R., Nunez, A. : 'Pipelined GaAs Carry Lookahead Adder', Electronics Letters, Sept. 1998, 34, (18), pp.1732 – 1733

[6] Cheng, Fu-Chiung, Stephen H. Unger, and Michael Theobald : 'Self-Timed Carry-Lookahead Adders', IEEE Transactions on Computers, 2000, 49, (7), pp. 659-672

[7] Aggoun, A., Ashur, A., and Ibrahim, M.K. : 'Novel Cell Architecture for High Performance Digit-Serial Computation', Electronics Letters, May 1993, 29, (11), pp. 938 – 940

[8] Nagendra, Irwin, and Owens: 'Area Time Power Tradeoffs in Parallel Adders', IEEE Trans. on Circuits and Systems, 1996, 43, (10), pp. 689 -702

[9] Koc, C.: 'RSA Hardware Implementation', RSA Laboratories, RSA Data Security, Inc. 1996

[10] Brent R. P. and H. T. Kung: 'A Regular Layout for Parallel Adders', IEEE Transactions on Computers, 1982, C-31, pp. 260-264

[11] Gutub, Adnan Abdul-Aziz, and Tahhan, Hassan, : 'Improving Cryptographic Architectures by Adopting Efficient Adders in their Modular Multiplication Hardware', The 9th Annual Gulf Internet Symposium, Oct. 2003, Khobar, Saudi Arabia.