Basic Adder Structures and the Lookahead Concept

James E. Stine
Electrical and Computer Engineering Department
Oklahoma State University
Stillwater, OK 74078, USA

1. Binary Addition and Subtraction Operations
   (a) In fixed point number systems, the simplest arithmetic operations are addition and subtraction.
   (b) To compute the sum of two 2's complement numbers ($S = A + B$), add the corresponding bits of both numbers. For example,

$$\begin{aligned} A &= 0.1011 = 11/16 \\ +B &= 0.0011 = 3/16 \\ = S &= 0.1110 = 14/16 \end{aligned}$$

   and

$$\begin{aligned} A &= 0.1001 = 9/16 \\ +B &= 1.0100 = -12/16 \\ = S &= 1.1101 = -3/16 \end{aligned}$$

   (c) To compute the difference of two 2's complement numbers ($S = A - B$), add the bits of the minuend A to the 2's complement of the subtrahend B. For example,

$$\begin{aligned} A &= 0.1011 = 11/16 \\ -B &= 0.0011 = 3/16 \end{aligned}$$

   becomes

$$\begin{aligned} A &= 0.1011 = 11/16 \\ +(-B) &= 1.1101 = -3/16 \\ = S &= 0.1000 = 8/16 \end{aligned}$$

2. Half Adders
   (a) A fundamental building block in arithmetic systems is the half adder (HA).
   (b) A HA takes two bits a_k and b_k and produces a sum bit s_k and a carry bit c_{k+1}.
   (c) The logic equations for a HA are:

$$\begin{aligned} s_k &= \overline{a_k} \cdot b_k + a_k \cdot \overline{b_k} \\ &= a_k \oplus b_k \\ c_{k+1} &= a_k \cdot b_k \end{aligned}$$

a_k	b_k	c_{k+1}	s_k
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Table 1: HA Truth Table

   (d) One possible logic implementation of the HA is shown in Figure 2d This HA requires 4 logic gates and has the following delays:

$$\begin{aligned} a_k, b_k \to s_k &= 3\,\triangle \\ a_k, b_k \to c_{k+1} &= 1\triangle \end{aligned}$$

   where $\triangle$ represents a single gate delay.

   (e) Full Adders
      i. A second building block in arithmetic systems is the full adder (FA).
      ii. A FA takes three bits a_k, b_k, and c_k and produces a sum bit s_k and a carry bit c_{k+1}.
      iii. The truth table for a FA is:

a_k	b_k	c_k	c_{k+1}	s_k
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Table 2: FA Truth Table

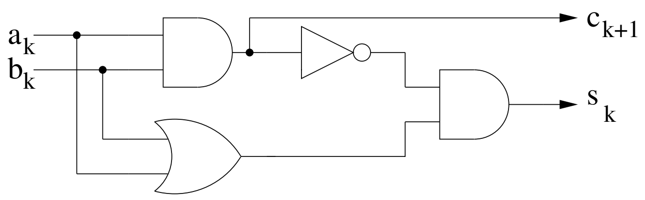

Figure 1: HA Implementation.

iv. The logic equations for a FA are:

$$\begin{aligned} s_k &= \overline{a_k}\cdot\overline{b_k}\cdot c_k + \overline{a_k}\cdot b_k\cdot\overline{c_k} + \\ &\quad a_k\cdot\overline{b_k}\cdot\overline{c_k} + a_k\cdot b_k\cdot c_k \\ &= a_k \oplus b_k \oplus c_k \\ c_{k+1} &= \overline{a_k}\cdot b_k\cdot c_k + a_k\cdot\overline{b_k}\cdot c_k + \\ &\quad a_k\cdot b_k\cdot\overline{c_k} + a_k\cdot b_k\cdot c_k \\ &= a_k\cdot b_k + a_k\cdot c_k + b_k\cdot c_k \end{aligned}$$

v. A FA can be constructed from two HAs and one OR gate, as shown in Figure 2(e)v. This FA requires 9 logic gates and has the following delays:

$$\begin{aligned} a_k, b_k \to s_k &= 6\,\triangle \\ a_k, b_k \to c_{k+1} &= 5\,\triangle \\ c_k \to s_k &= 3\,\triangle \\ c_k \to c_{k+1} &= 2\,\triangle \end{aligned}$$

vi. Ripple Carry Adders

A. Ripple carry adders (RCA) provides a slow, but hardware efficient, method for adding two binary numbers.

B. An n-bit RCA is formed by concatenating n FAs.

C. The carry out from the k^{th} FA is used as the carry in of the $(k+1)^{th}$ FA, as shown in Figure 2(e)viC.

D. Since an n-bit RCA requires n FAs and each FA has nine gates, the total number of gates for the n-bit RCA is $9n$. The worst case delays for a ripple carry adder are:

$$\begin{aligned} a_0, b_0 \to s_{n-1} &= (2n+4)\,\triangle \\ a_0, b_0 \to c_n &= (2n+3)\triangle \end{aligned}$$

vii. Ripple Carry Adders/Subtracters

A. To perform subtraction, it is necessary to complement each of the bits of B and add a one to the least significant bit. This is accomplished by adding a row of n XOR gates to form a ripple-carry adder/subtractor (RCSA), as shown in Figure 2(e)viiA.

B. To perform subtraction, *sub* is set to one, which complements the bits of B and sets c_0 to one. To perform addition, *sub* is set to zero, which leaves the bits of B unchanged and sets c_0 to zero.

C. If an XOR gate is equivalent to 4 simple gates, and has a delay of $3\triangle$, the RSCA requires $13n$ gates and has the following worst case delays:

$$\begin{aligned} b_0 \to s_{n-1} &= (2n+7)\,\triangle \\ b_0 \to c_n &= (2n+6)\triangle \end{aligned}$$

viii. Overflow Detection

A. Overflow occurs when you compute an operation with two valid representations and the result cannot be represented for the given representation because the value is too large or too small.

B. Specific detection of overflow requires knowing which operation is being performed and its given representation (e.g., two's complement).

C. For an unsigned number representation, a simple way of detecting overflow is add two numbers and check whether the result is larger than the largest posible value (e.g., for an n-bit integer this would be $2^n - 1$). This is because it would take $n+1$ bits to represent the result correctly.

D. If $A+B$ or $A-B$ is greater than or equal to $+1$, or less than -1, overflow occurs and the computed sum

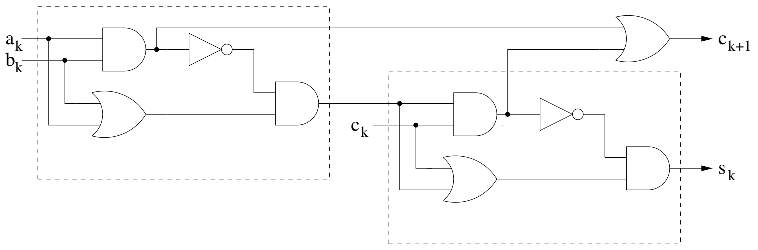

Figure 2: FA Implementation.

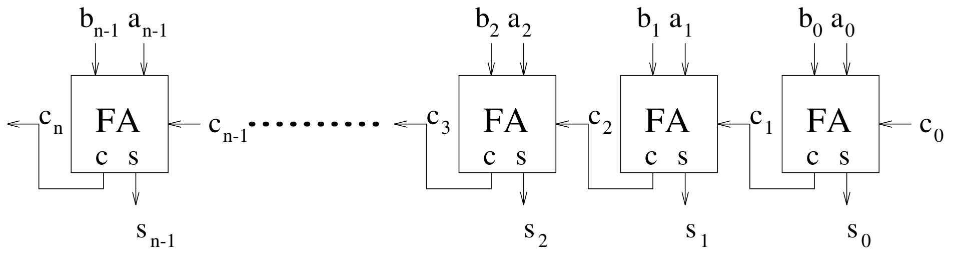

Figure 3: RCA Implementation.

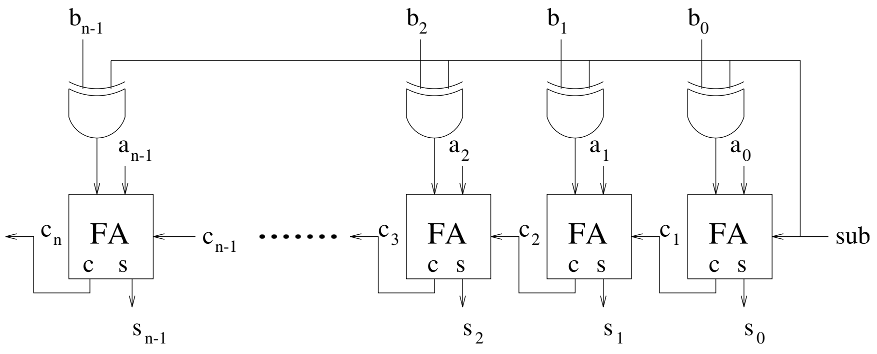

Figure 4: RCAS Implementation.

or difference is incorrect. For example,

$$\begin{aligned} A &= 0.1001 = 9/16 \\ +B &= 0.1100 = 12/16 \\ = S &= 1.0101 = -13/16 \end{aligned}$$

is an incorrect result

E. One way to detect overflow is to check the result of the sum. However, this can be complicated because as the input operand grows, it requires the result before it can determine an overflow, which is computationally complex.

F. A simple method to detect overflow by comparing the the carry into the most significant bit to the carry out of the most significant bit. If these two bits differ, then overflow has occurred. This can be expressed as:

$$OV = c_{n-1} \oplus c_n$$

For example, in the previous computation of $A = 9/16$ added to $B = 12/16$:

$$\begin{aligned} c_{n-1} &= 1 \\ c_n &= 0 \\ OV &= 1 \oplus 0 = 1 \end{aligned}$$

and overflow is detected.

(a) Carry Lookahead Concept.

i. A carry lookahead adder uses specialized logic to compute the carries in parallel.

ii. A full adder generates a carry if both a_k and b_k are one.

$$g_k = a_k \cdot b_k$$

iii. A full adder propagates a carry if either a_k or b_k is one.

$$p_k = a_k + b_k \tag{1}$$

iv. For example, when adding the numbers

$$\begin{aligned} A &= 0.1010 \\ +B &= 0.0011 \end{aligned}$$

carries are generated in position 1 and propagated in positions 0 and 3.

(b) Carry Lookahead Equations.

i. A carry enters position $k+1$ if a carry is generated in position k or a carry enters position k and is propagated.

$$c_{k+1} = g_k + p_k \cdot c_k$$

ii. Solving this equation recursively gives:

$$\begin{aligned} c_{k+2} &= g_{k+1} + p_{k+1} \cdot c_{k+1} \\ &= g_{k+1} + p_{k+1} \cdot (g_k + p_k \cdot c_k) \\ &= g_{k+1} + p_{k+1} \cdot g_k + p_{k+1} \cdot p_k \cdot c_k \end{aligned}$$

iii. Extending this to one more position gives:

$$\begin{aligned} c_{k+3} &= g_{k+2} + p_{k+2} \cdot c_{k+2} \\ &= g_{k+2} + p_{k+2} \cdot \\ &\quad (g_{k+1} + p_{k+1} \cdot g_k + \ p_{k+1} \cdot p_k \cdot c_k) \\ &= g_{k+2} + p_{k+2} \cdot g_{k+1} + \\ &\quad p_{k+2} \cdot p_{k+1} \cdot g_k + \\ &\quad p_{k+2} \cdot p_{k+1} \cdot p_k \cdot c_k \end{aligned}$$

(c) Generalized Carry Lookahead Equations.

i. The equation for the carry into position $k + r$ is:

$$c_{k+r} = \left(\sum_{i=k}^{k+r-1} g_i \prod_{j=i+1}^{k+r-1} p_j \right) + c_k \prod_{j=k}^{k+r-1} p_j$$

ii. Implementing this equation requires $r+1$ gates, which have a maximum fan-in of $r + 1$.

iii. To produce all the carries from c_{k+1} to c_{k+r} a total of

$$\sum_{i=1}^{r} (i+1) = \frac{(r+3) \cdot r}{2}$$

gates are required.

(d) Modification of the original 9 gate FA.

i. The 9 gate full adder already has a the required logic to produce g_k and p_k.

ii. In addition, the OR gate used to produce c_{k+1} is no longer required.

iii. Consequently, we can use a reduced full adder (RFA) that requires only 8 logic gates.

iv. The generate and propagate signals are ready after $1\triangle$

(e) A 4-bit CLA.

i. For $k = 0$ we have:

$$\begin{aligned} c_1 &= g_0 + p_0 \cdot c_0 \\ c_2 &= g_1 + p_1 \cdot g_1 + p_1 \cdot p_0 \cdot c_0 \\ c_3 &= g_2 + p_2 \cdot g_1 + p_2 \cdot p_1 \cdot g_0 + \\ &\quad p_2 \cdot p_1 \cdot p_0 \cdot c_0 \end{aligned}$$

The logic used to produce the carries is typically referred to as a carry lookahead generator (CLG).

ii. Draw figure of a 4-bit CLA.

iii. A 4-bit CLG uses 9 gates and has two gate delays. However, some of these gates have more than two inputs.

iv. Since a 4-bit CLA uses 4 RFAs, and a 4-bit CLG (9 gates), it has a total of $4 \cdot 8 + 9 = 41$ gates.

v. The worst case delay for a 4-bit CLA is:

$$\begin{aligned} a_k, b_k \to s_3 &= 6\,\triangle \\ a_k, b_k \to c_4 &= 5\,\triangle \end{aligned}$$

(f) Block Carry Lookahead Generators

i. Each additional position, increases the fan-in of the logic gates. Most technologies have a maximum fan-in of four. Therefore, to continue the process, it is necessary to define generate and propagate signals over four bit blocks.

$$\begin{aligned} g_{k+3:k} &= g_{k+3} + p_{k+3} \cdot g_{k+2} + \\ &\quad p_{k+3} \cdot p_{k+2} \cdot g_{k+1} + \\ &\quad p_{k+3} \cdot p_{k+2} \cdot p_{k+1} \cdot g_k \\ p_{k+3:k} &= p_{k+3} \cdot p_{k+2} \cdot p_{k+1} \cdot p_k \end{aligned}$$

ii. A four bit block carry lookahead generator (BCLG) has 14 gates, and a maximum delay of $2\triangle$.

iii. In terms of the four bit block generate and propagate signals, the next carry can be expressed as

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

iv. A $4 - bit$ CLA with block generate and propagate signals requires 46 gates.

(g) Larger Carry Lookahead Adders

i. Draw 16-bit CLA.

ii. A 16-bit CLA requires 16 RFAs (8 gates each) and 5 CLA blocks (14 gates each). Therefore, it requires a total of $16 \cdot 8 + 5 \cdot 14 = 198$ gates.

iii. The delays through the 16-bit CLA are:

$$\begin{aligned} a_k, b_k \rightarrow p_k, g_k &= 1 \triangle \\ p_k, g_k \rightarrow g_{k+3:k} &= 2 \triangle \\ g_{k+3:k} \rightarrow c_{k+4} &= 2 \triangle \\ c_{k+4} \rightarrow c_{k+7} &= 2 \triangle \\ c_{k+7} \rightarrow s_{k+7} &= 3 \triangle \\ a_k, b_k \rightarrow c_{k+7} &= 10 \triangle \end{aligned}$$

(h) CLA Gate Counts

i. An n-bit CLA with a maximum fan-in of r, requires

$$\sum_{l=1}^{log_r(n)} \frac{n}{r^l} \approx \lceil \frac{n-1}{r-1} \rceil$$

carry lookahead blocks and n RFAs.

ii. An r-bit carry lookahead block requires $\frac{(3+r)r}{2}$ gates and each RFA requires 8 gates.

iii. Thus, the total number of gates for an n-bit CLA is

$$8n + \frac{(3+r)r}{2} \cdot \lceil \frac{n-1}{r-1} \rceil$$

iv. When $r = 4$, this gives

$$8n + 14 \cdot \lceil \frac{n-1}{3} \rceil \approx 12.67n - 4.67$$

v. When $r = 2$, this gives

$$8n + 5(n-1) = 13n - 5$$

(i) CLA Delay

i. An n-bit CLA with a maximum fan-in of r, requires $\lceil \log_r(n) \rceil$ CLA logic levels.

ii. An r-bit CLA has six gate delays and there are four additional gate delays per level after the first.

iii. Thus, the delay for an n-bit CLAs is

$$6 + 4 \cdot (\lceil \log_r(n) \rceil - 1) = \; 2 + 4 \cdot \lceil \log_r(n) \rceil$$

(j) Compared to the RCA, the CLA:

i. Has a much shorter delay (logarithmic instead of linear).

ii. Uses a larger number of gates.

iii. Is less regular in structure.

(k) Carry Lookahead Adder Block Issues

i. Fan-out

ii. Logic Levels

iii. Wire Tracks (density)

(l) Quantitative Comparison

Size	CLA	CLA	RCA
n	$(r = 4)$	$(r = 2)$	
4	46	47	36
8	92	99	72
16	198	203	144
32	396	411	288
64	806	827	576
128	1,612	1,659	1,152

Table 3: Area

Size	CLA	CLA	RCA
n	$(r = 4)$	$(r = 2)$	
4	6	10	12
8	10	14	20
16	10	18	36
32	14	22	68
64	14	26	132
128	18	30	260

Table 4: Delay

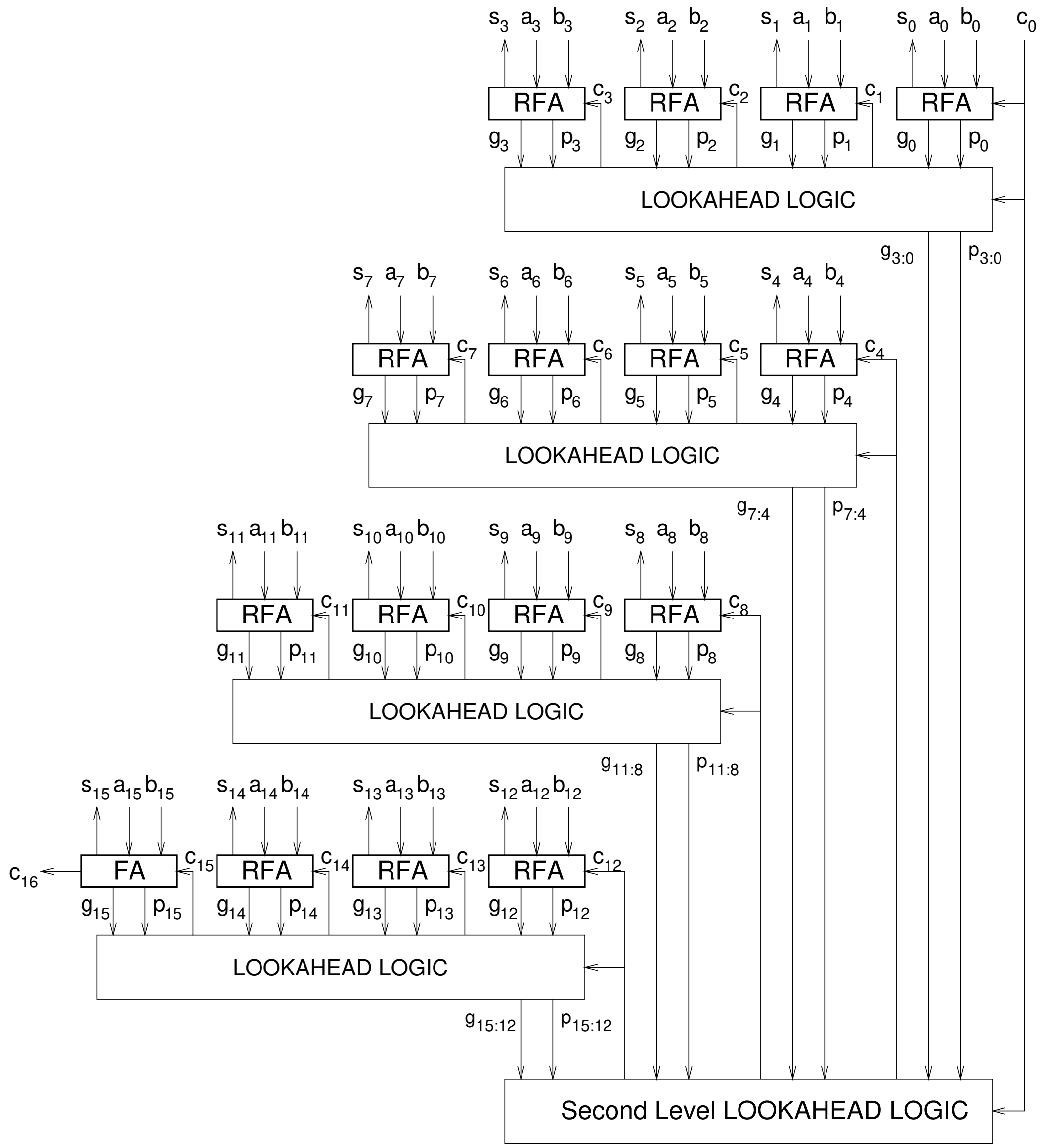
s3 a3 b3 s2 a2 b2 s1 a1 b1 s0 a0 b0 c0
RFA c3 RFA c2 RFA c1 RFA
g3 p3 g2 p2 g1 p1 g0 p0
LOOKAHEAD LOGIC
g3:0 p3:0
s7 a7 b7 s6 a6 b6 s5 a5 b5 s4 a4 b4
RFA c7 RFA c6 RFA c5 RFA c4
g7 p7 g6 p6 g5 p5 g4 p4
LOOKAHEAD LOGIC
g7:4 p7:4
s11 a11 b11 s10 a10 b10 s9 a9 b9 s8 a8 b8
RFA c11 RFA c10 RFA c9 RFA c8
g11 p11 g10 p10 g9 p9 g8 p8
LOOKAHEAD LOGIC
g11:8 p11:8
s15 a15 b15 s14 a14 b14 s13 a13 b13 s12 a12 b12
c16 FA c15 RFA c14 RFA c13 RFA c12
g15 p15 g14 p14 g13 p13 g12 p12
LOOKAHEAD LOGIC
g15:12 p15:12
Second Level LOOKAHEAD LOGIC

Figure 5: 16-bit Carry Lookahead Adder.