Majority-Based Decomposition of Carry Logic in Binary Adders

Leyla Nazhandali and Karem A. Sakallah

CSE-TR-451-02

February 2002

THE UNIVERSITY OF MICHIGAN

Computer Science and Engineering Division
Department of Electrical Engineering and Computer Science
Ann Arbor, Michigan 48109-2122
USA

Majority-Based Decomposition of Carry Logic in Binary Adders

Leyla Nazhandali and Karem A. Sakallah

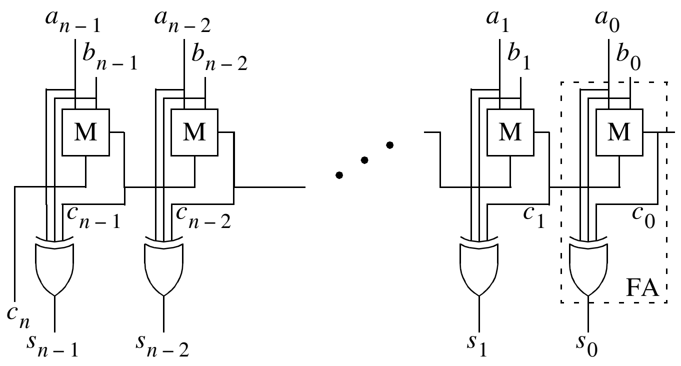
Advanced Computer Architecture Laboratory
Department of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor, Michigan 48109-2122

February 2002

Abstract

We introduce a new carry lookahead adder architecture based on a decomposition of the carry logic into a network consisting exclusively of majority gates. We call adders with this architecture "M&M" adders to distinguish them from adders based on the conventional Generate/Propagate, or "G&P," architecture. We show how M&M and G&P adders are related, and characterize optimal realizations of M&M adders.

Notational Conventions



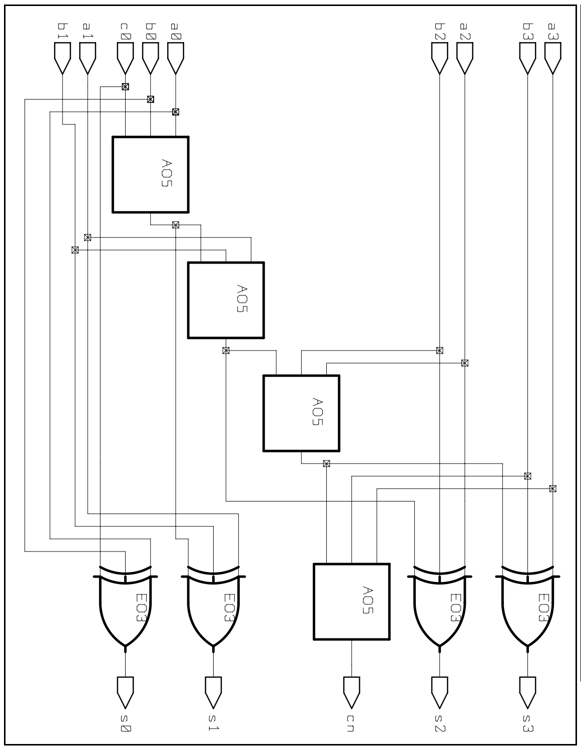
We designate the augend, addend, and sum of an n-bit adder as $A = \langle a_{n-1}, \ldots, a_1, a_0 \rangle$, $B = \langle b_{n-1}, \ldots, b_1, b_0 \rangle$, and $S = \langle s_{n-1}, \ldots, s_1, s_0 \rangle$ (see figure.) The carry into bit position i is labeled c_i; thus, the initial carry (into bit position 0) is c_0 and the final carry (out of bit position $n-1$) is c_n. The sum and carry outputs at bit position i obey the following "full adder" equations:

$$\begin{aligned} s_i &= a_i \oplus b_i \oplus c_i \\ c_{i+1} &= a_i b_i + (a_i + b_i) c_i \equiv \mathrm{MAJ}(a_i, b_i, c_i) \end{aligned} \tag{0.1}$$

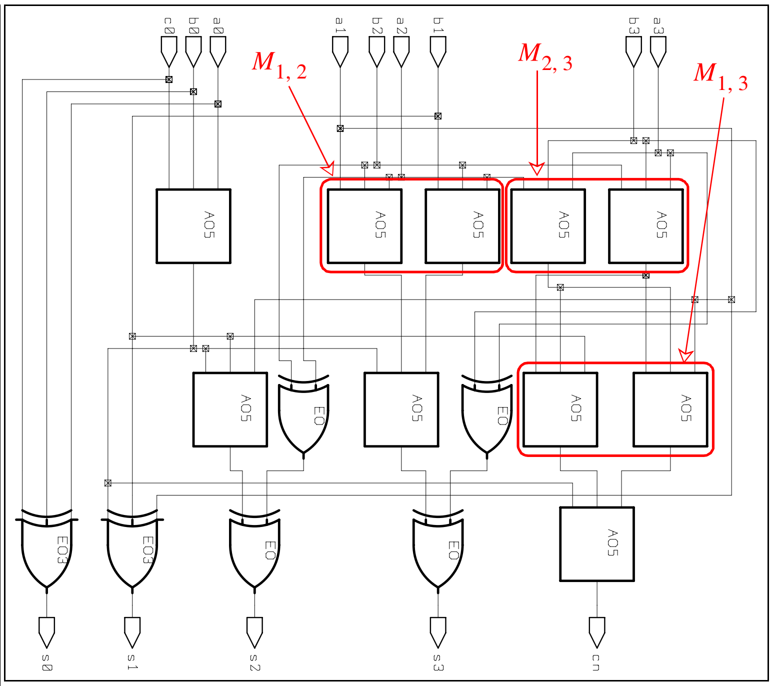
where "MAJ" is used to denote the 3-input majority gate. Schematically, we will draw a majority gate as a box labeled with M. Finally, we note that even though MAJ is symmetric in all three inputs, we will graphically distinguish the carry input by having it enter the gate from its right, as opposed to its top, side to more closely mimic carry propagation on positional numbers.

1 Introduction

Adders have been studied extensively because of their central importance in the design of digital hardware. Classical high speed adders include carry lookahead, carry skip [6], carry select [1], and conditional sum [7] adders. Most high speed adders, such as the well-known Brent and Kung adder [2], utilize some form of prefix computation [5] to accelerate the generation of carry bits. Such *carry lookahead* (CLA) is almost exclusively based on a recursive decomposition of the carry logic to yield a multi-level circuit—whose depth is logarithmic in n—that computes *generate* and *propagate* signals over contiguous bit groupings. Denoting the generate and propagate signals over the bit range $[i, k]$ by $G_{i,k}$ and $P_{i,k}$ (with $G_{i,i} \equiv a_i b_i$ and $P_{i,i} \equiv a_i + b_i$) allows the carry out of bit position k to be expressed as $c_{k+1} = G_{i,k} + P_{i,k} c_i$. This equation also "explains" the names chosen for these signals: $G_{i,k}$ is true when a carry is generated within the bit range $[i, k]$, and $P_{i,k}$ is true when an input carry c_i can be propagated across this bit range. We will refer to CLA adders based on generate and propagate signals as G&P adders.



(a) A ripple carry adder is produced when the least complex logic first (LCLF) heuristic is used



(b) An M&M carry lookahead adder is produced when the most complex logic first (MCLF) heuristic is used

Figure 1-1: 4-bit adders synthesized by M31. The AO5 “boxes” represent MAJ gates.

In this report we introduce a new decomposition of carry logic that is *not* based on generate and propagate signals. Rather, it is based on pairs of symmetric signals that are computed by a prefix network consisting of just one gate type, namely MAJ. We will denote such signals, over the bit range $[i, k]$, as $M^a_{i,k}$ and $M^b_{i,k}$ to reflect i) that they are the outputs of majority gates, and ii) that they come in pairs corresponding to the augend and addend of the adder. Furthermore, these signals reduce to augend and addend bits when $k = i$, i.e. $M^a_{i,i} \equiv a_i$ and $M^b_{i,i} \equiv b_i$. In terms of such signals, the carry out of bit position k is simply the output of another MAJ gate: $c_{k+1} = \mathrm{MAJ}(M^a_{i,k}, M^b_{i,k}, c_i)$. The “meaning” of these M signals can be inferred by applying the definition of MAJ in the above equation yielding $c_{k+1} = M^a_{i,k}M^b_{i,k} + (M^a_{i,k} + M^b_{i,k})c_i$. Thus, a carry is generated within the bit range $[i, k]$ when *both* M signals are true, whereas a carry is propagated across the bit range when *either* signal is true. Recalling that the carry function over a single bit is itself a majority operation, this equation can be viewed as a generalization of majority from a single bit to a bit range. We will refer to CLA adders based on such a decomposition as M&M adders.

The M&M structure was "discovered" by our logic synthesis tool M31 [3, 4]. M31 uses a constructive synthesis paradigm that builds a circuit implementation incrementally from the primary inputs towards the primary outputs. At each iteration, M31 decomposes the still-unimplemented forward logic to extract and instantiate a small multi-output subfunction corresponding to a few (one to four) fanin-limited primitive library gates. To insure progress, M31 chooses decompositions that lead to a reduction in circuit width, i.e. in the number of signals that are direct inputs to the remaining unimplemented logic. Logic functions that can be decomposed in this fashion must satisfy certain minimal regularity requirements, namely they must have at most 2^{s-1} distinct cofactors in terms of some s-sized subset of their inputs. M31 employs fast heuristics to identify such subsets and has several "knobs" to choose among candidate subsets. The schematics in Figure 1-1 are for 4-bit adders that were synthesized by M31 using two different decomposition orders. The structure in (a) is a ripple-carry adder that was obtained by choosing variable subsets according to the *least complex logic first* (LCLF) heuristic: for adders, this amounts to constructing the circuit implementation starting from the least-significant bit (LSB) and proceeding towards the most-significant bit (MSB.) The structure in (b), on the other hand, was obtained using the *most complex logic first* (MCLF) heuristic which performs the decomposition in the opposite (MSB to LSB) direction. Close examination of this circuit shows that the carry into the most significant bit position (i.e. c_3) is generated at a logic depth which is logarithmic in the adder width ($\log_2 4 = 2$). Thus, this circuit represents a new type of CLA structure in which the only gate type used is MAJ.

Discovery of this structure motivated the study described in this report. The rest of the report is organized as follows. In Sec. 2 we set the stage for the derivation of the M&M decomposition equations by proving the distributivity of MAJ. The fundamental M&M decomposition equations are introduced in Sec. 3. In Sec. 4 we use these equations to derive the architecture of minimum-depth M&M adders. We conclude, in sec 5, with a brief comparison between the G&P and M&M structures.

2 Distributivity of Majority

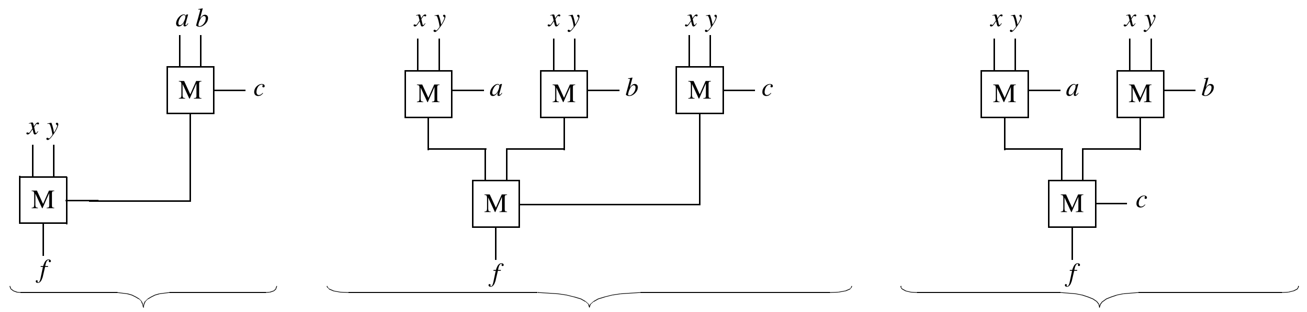
As will become clear in the sequel, the M&M structure arises because of the distributivity of the majority function over itself. This property is stated by the following theorem (see also Figure 2-1):

Theorem 2.1 For arbitrary Boolean variables a, b, c, x, and y we have

$$\mathrm{MAJ}(x, y, \mathrm{MAJ}(a, b, c)) = \mathrm{MAJ}(\mathrm{MAJ}(x, y, a), \mathrm{MAJ}(x, y, b), \mathrm{MAJ}(x, y, c)) \tag{2.1}$$

representing the distribution of the outer (second-level) majority over the inner (first-level) majority.

Proof: Expand the right side of (2.1) using the definition of majority from (0.1), then manipulate using idempotence and distributivity of AND over OR and OR over AND:



$$\text{MAJ}(x, y, \text{MAJ}(a, b, c)) = \text{MAJ}(\text{MAJ}(x, y, a), \text{MAJ}(x, y, b), \text{MAJ}(x, y, c)) = \text{MAJ}(\text{MAJ}(x, y, a), \text{MAJ}(x, y, b), c)$$

(i) (ii) (iii)

Figure 2-1: Distributivity of majority

$$\begin{aligned}
&\text{MAJ}(\text{MAJ}(x, y, a), \text{MAJ}(x, y, b), \text{MAJ}(x, y, c)) \\
&\quad = \text{MAJ}([xy + (x+y)a], [xy + (x+y)b], [xy + (x+y)c]) \\
&\quad = [xy + (x+y)a][xy + (x+y)b] + \{[xy + (x+y)a] + [xy + (x+y)b]\}[xy + (x+y)c] \\
&\quad = [xy + (x+y)ab] + [xy + (x+y)(a+b)][xy + (x+y)c] \\
&\quad = [xy + (x+y)ab] + [xy + (x+y)(a+b)c] \\
&\quad = xy + (x+y)(ab + (a+b)c) \\
&\quad = xy + (x+y)\text{MAJ}(a, b, c) \\
&\quad = \text{MAJ}(x, y, \text{MAJ}(a, b, c)) \qquad \blacksquare
\end{aligned}$$

A variation of this identity, in which the outer majority is distributed over only two, instead of all three, inputs to the inner majority, is also easy to establish. Specifically,

$$\text{MAJ}(x, y, \text{MAJ}(a, b, c)) = \text{MAJ}(\text{MAJ}(x, y, a), \text{MAJ}(x, y, b), c) \tag{2.2}$$

This is shown schematically in part (iii) of Figure 2-1. It is instructive to note that, due to the symmetry of the majority function, the transformation of the 2-gate circuit in part (i) to the 3-gate circuit in part (iii) is not unique. In particular, the inputs of the first-level MAJ in part (i) can be assigned arbitrarily to the three MAJ gates in part (iii). To be useful in CLA structures, however, we must choose the assignment that reduces overall circuit depth. As we shall see, if a and b represent augend and addend (groups of) bits and c represents a carry signal, such an assignment corresponds to what is shown in part (iii), where the carry signal is assigned to the second-level majority gate.

3 The M&M CLA Architecture

The M&M decomposition of carry logic is based on the following recursive definition of pairs of symmetric signals that, together, capture the carry generation and propagation properties of the augend and addend bits:

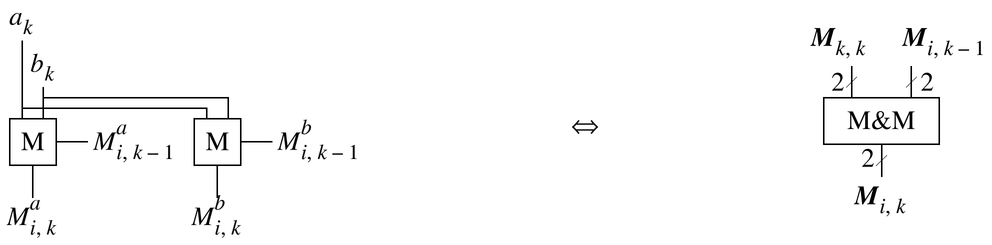


Figure 3-1: Double majority cell for computing a pair of M signals. Note that $M_{k,k} = \langle a_k, b_k \rangle$

Definition 3.1 (M Signals) M signals are defined in pairs as follows:

$$\begin{aligned} &M^a_{i,i} \equiv a_i && M^b_{i,i} \equiv b_i && \\ &M^a_{i,k} \equiv \mathrm{MAJ}(a_k, b_k, M^a_{i,k-1}) && M^b_{i,k} \equiv \mathrm{MAJ}(a_k, b_k, M^b_{i,k-1}) && 0 \le i < k < n \end{aligned} \tag{3.1}$$

In these definitions, the subscript of an M signal indicates the bit range over which it is computed, and its superscript denotes the adder argument (either a or b) to which it corresponds. Since they occur in pairs, we will sometimes use the shorthand $\boldsymbol{M}_{i,k}$ to represent both signals. Formally,

$$\boldsymbol{M}_{i,k} \equiv \langle M^a_{i,k}, M^b_{i,k} \rangle \qquad 0 \le i \le k < n \tag{3.2}$$

It will also be convenient to define the range R of an M signal as the number of bits over which it is computed. Thus,

$$R(\boldsymbol{M}_{i,k}) = k - i + 1 \tag{3.3}$$

Starting from M signals with a range of one bit, namely the adder inputs, the recursive definition in (3.1) provides the basic mechanism for extending the range of an M signal pair by a single bit. This is shown schematically in Figure 3-1. As we will shortly see, the "double majority" cell used in this construction is the basic building block in the M&M CLA architecture.

The relation between carry and M signals is provided by the following theorem:

Theorem 3.1 (Carry Generation): Given a bit range $[i, k]$, the carry out of bit position k can be obtained from $\boldsymbol{M}_{i,k}$ and the carry into bit position i according to:

$$c_{k+1} = \mathrm{MAJ}(M^a_{i,k}, M^b_{i,k}, c_i) \qquad i \le k \tag{3.4}$$

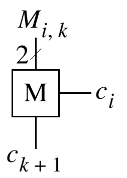


Figure 3-2: Carry generation

Proof: (Basis) Equation (3.4) holds for $k = i$:

$$c_{i+1} = \mathrm{MAJ}(a_i, b_i, c_i) \qquad \text{[Definition of } c_{i+1} \text{ from (0.1)]}$$

$$= \mathrm{MAJ}(M^a_{i,i}, M^b_{i,i}, c_i) \qquad \text{[Definition of } M_{i,i} \text{ from (3.1)]}$$

(Induction) Suppose that (3.4) holds for some $k-1 \ge i$, i.e. $c_k = \mathrm{MAJ}(M^a_{i,k-1}, M^b_{i,k-1}, c_i)$. Then,

$$\mathrm{MAJ}(M^a_{i,k}, M^b_{i,k}, c_i) = \mathrm{MAJ}(\mathrm{MAJ}(a_k, b_k, M^a_{i,k-1}), \mathrm{MAJ}(a_k, b_k, M^b_{i,k-1}), c_i) \qquad \text{[Definition of } M_{i,k} \text{ from (3.1)]}$$

$$= \mathrm{MAJ}(a_k, b_k, \mathrm{MAJ}(M^a_{i,k-1}, M^b_{i,k-1}, c_i)) \qquad \text{[By Theorem 2.1]}$$

$$= \mathrm{MAJ}(a_k, b_k, c_k) \qquad \text{[Induction assumption]}$$

$$= c_{k+1} \qquad \text{[Definition of carry from (0.1)]}$$

Thus, by induction, (3.4) holds for all $k \ge i$. ■

It can be easily seen from the above theorem that the index of generated carry is related to the index of input carry and the range of M signal as in (3.5):

$$k + 1 = i + R(\boldsymbol{M}_{i,k}) \tag{3.5}$$

Definition (3.1) tells us that an M signal that ranges over $(k - i + 1)$ bits can be constructed from two “lower level” M signals that range over $(k - i)$ bits and one bit, respectively. This construction can be generalized so that an M signal can be obtained from any two lower level M signals with contiguous bit ranges as long as the sum of their ranges equals that of M. This is formalized in the following theorem.

Theorem 3.2 (M Signal Generation): Given two contiguous bit ranges $[i, j]$ and $[j+1, k]$, the M signals over the combined range $[i, k]$ can be computed from the M signals over the sub-ranges according to:

$$\begin{aligned} M^a_{i,k} &= \mathrm{MAJ}(M^a_{j+1,k}, M^b_{j+1,k}, M^a_{i,j}) \\ M^b_{i,k} &= \mathrm{MAJ}(M^a_{j+1,k}, M^b_{j+1,k}, M^b_{i,j}) \end{aligned} \qquad i \le j < k < n \tag{3.6}$$

Proof: We use induction to show that (3.6) holds for $M^a_{i,k}$; symmetry establishes it for $M^b_{i,k}$.

(Basis) For any i and j such that $0 \le i \le j < n - 1$, let $k = j + 1$. Thus,

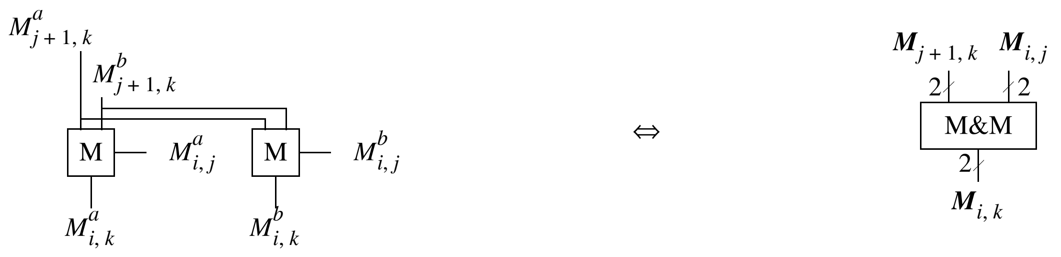


Figure 3-3: M signal generation

$$M^a_{i,j+1} = \mathrm{MAJ}(a_{j+1}, b_{j+1}, M^a_{i,j}) \qquad \text{[Definition of } M^a_{i,j+1} \text{ from (3.1)]}$$

$$= \mathrm{MAJ}(M^a_{j+1,j+1}, M^b_{j+1,j+1}, M^a_{i,j}) \qquad \text{[Definition of } \boldsymbol{M}_{j+1,j+1} \text{ from (3.1)]}$$

(Induction) Suppose that (3.6) holds for some $k-1 \geq j+1$, i.e. $M^a_{i,k-1} = \mathrm{MAJ}(M^a_{j+1,k-1}, M^b_{j+1,k-1}, M^a_{i,j})$. Then,

$$\mathrm{MAJ}(M^a_{j+1,k}, M^b_{j+1,k}, M^a_{i,j})$$

$$= \mathrm{MAJ}(\mathrm{MAJ}(a_k, b_k, M^a_{j+1,k-1}), \mathrm{MAJ}(a_k, b_k, M^b_{j+1,k-1}), M^a_{i,j}) \qquad \text{[Definition of } \boldsymbol{M}_{j+1,k} \text{ from (3.1)]}$$

$$= \mathrm{MAJ}(a_k, b_k, \mathrm{MAJ}(M^a_{j+1,k-1}, M^b_{j+1,k-1}, M^a_{i,j})) \qquad \text{[By Theorem 2.1]}$$

$$= \mathrm{MAJ}(a_k, b_k, M^a_{i,k-1}) \qquad \text{[Induction assumption]}$$

$$= M^a_{i,k} \qquad \text{[Definition of } M^a_{i,k} \text{ from (3.1)]}$$

Thus, by induction, (3.6) holds for all $j < k < n$. ■

It will be convenient, in some instances, to represent the two scalar equations in (3.6) by the single vector equation:

$$\boldsymbol{M}_{i,k} = \mathrm{MMAJ}(\boldsymbol{M}_{i,j}, \boldsymbol{M}_{j+1,k}) \qquad (3.7)$$

where $\boldsymbol{M}_{i,j}$, $\boldsymbol{M}_{j+1,k}$, and $\boldsymbol{M}_{i,k}$ are defined as in (3.2), and "MMAJ" is used to denote the double-majority gate (see Figure 3-3). We also note, for future reference, that the range of the output M signals is related to those of the input M signals according to:

$$R(\boldsymbol{M}_{i,k}) = R(\boldsymbol{M}_{i,j}) + R(\boldsymbol{M}_{j+1,k}) \qquad (3.8)$$

Theorem 3.1 and Theorem 3.2 provide the necessary machinery for generating all possible M&M decompositions of carry logic. Using these two theorems, a given carry can be generated from any lower carry and corresponding M signals yielding a large number of possible decompositions that differ in their size, depth, and fan-in/fan-out characteristics. The maximum depth M&M adder turns out to be the canonical ripple structure. In the following section we examine the structure of minimum-depth M&M adders.

4 Minimum-Depth M&M Adders

Of the many possible M&M adder structures that can be created in accordance with Theorem 3.1 and Theorem 3.2, we focus in this section on those that have the fewest number of logic levels, i.e. those that are of minimum depth. Assuming that the logic level[1] of the adder inputs is 0, the logic level of a carry or an M signal generated using the constructors in (3.4) and (3.6) will be one more than the maximum level of any of its arguments. We will show that minimum-depth adders have a unique *spine* that generates carry signals $c_1, c_3, c_{7,\ \dots}$, i.e. carry signals whose index is $2^l - 1$ for $l \geq 1$. Each of the remaining carry signals can be generated in several ways that offer trade-offs between area and fanout requirements. The characteristics of minimum-depth M&M adders are established by the following two lemmas.

Lemma 4.1 The maximum range of any M signal that can be generated at a given level l, such that $0 \leq l \leq \log_2 n$, is 2^l.

Proof: We establish the proof using induction.

(Basis) The only M signals at level 0 are $\boldsymbol{M}_{i,\,i}$ for $0 \leq i < n$. Since $R(\boldsymbol{M}_{i,\,i}) = 1 = 2^0$, the lemma obviously holds for $l = 0$.

(Induction) Suppose the lemma is true for all levels from 0 to some arbitrary level l, and consider the generation of an M signal $\boldsymbol{M}_{i,\,k}$ at level $l + 1$ according to (3.7). From (3.8), the range of this signal is maximized when the ranges of its constituent signals $\boldsymbol{M}_{i,\,j}$ and $\boldsymbol{M}_{j+1,\,k}$ are individually maximized. Since these signals must necessarily be generated at levels that are lower than $l + 1$, they attain their maximum ranges when they are generated at level l, namely 2^l. This immediately leads to a maximum range of $\boldsymbol{M}_{i,\,k}$ at level $l + 1$ of $2^l + 2^l = 2^{l+1}$, thus establishing the validity of the lemma for $l + 1$ and proving it for all l by induction. We should note that the upper bound on l insures that the maximum M signal range does not exceed the adder width n. ■

Corollary 4.2 Given an M signal with range R, the lowest level at which it can be generated is $\lceil \log_2 R \rceil$.

Lemma 4.3 The "highest" carry, i.e. that carry with the largest index, that can be generated at level $l \geq 0$ is $c_{2^l - 1}$.

1. Note that we use "depth" and "level" synonymously.

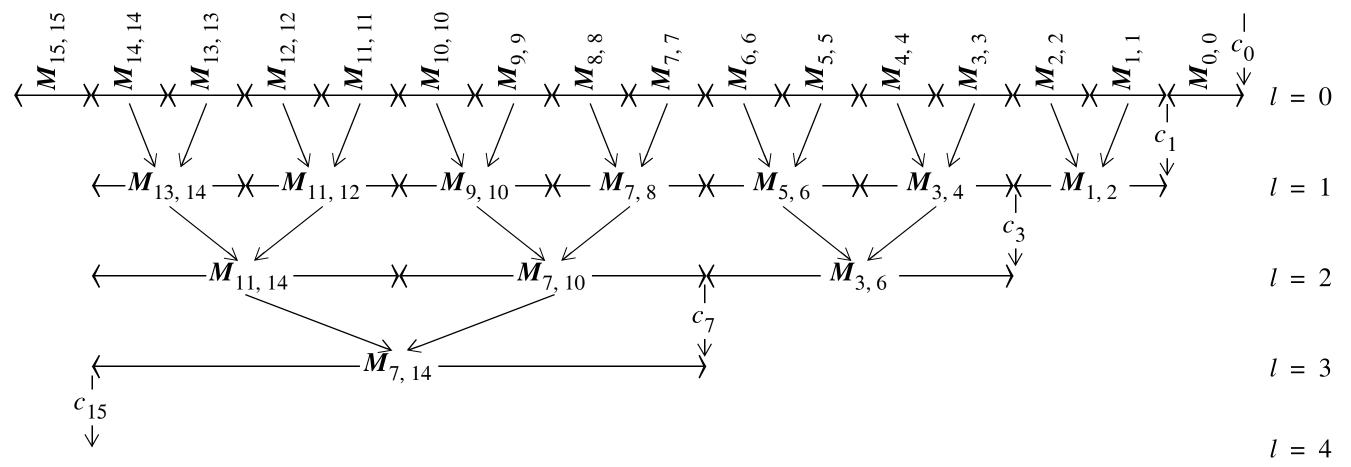


Figure 4-1: Spine of a minimum-depth 16-bit M&M adder

Proof: We establish the proof using induction.

(Basis) The highest (and only) carry at level 0 is c_0. Since, $2^0 - 1 = 0$, the lemma obviously holds for $l = 0$.

(Induction) Suppose the lemma is true for all levels from 0 to some arbitrary level l, and consider the generation of a carry signal c_{k+1} at level $l+1$ according to (3.4). From (3.5), the index of this signal is maximized when the index of c_i and the range of $\boldsymbol{M}_{i,k}$ are individually maximized. Since these signals must necessarily be generated at levels that are lower than $l+1$, they attain their maximum index and range, respectively, when they are generated at level l. From the induction hypothesis, this yields a maximum index of $2^l - 1$ for c_i; and from Lemma 4.1, we obtain a maximum range of 2^l for $\boldsymbol{M}_{i,k}$. The maximum index of c_{k+1} at level $l+1$, thus, is $2^l - 1 + 2^l = 2^{l+1} - 1$, establishing the validity of the lemma for $l+1$ and proving it for all l by induction. ■

Corollary 4.4 The lowest level at which a carry c_k can be generated is $\lceil \log_2(k+1) \rceil$, and the minimum depth of an n-bit M&M adder is $1 + \lceil \log_2(n) \rceil$ (the lowest level at which c_{n-1} can be generated plus one more level to generate s_{n-1}.)

The spine of a minimum-depth 16-bit M&M adder, showing all of the required M signals, is illustrated in Figure 4-1. In this figure, M signals are depicted as horizontal intervals over their respective ranges, and the double majority gates needed to create them are shown as slanted downward-pointing arrows. For an n-bit adder with $n = 2^m$, it is easy to establish that the number of double majority gates in the spine is $2^m - (m+1)$ 2.

2. Note that the spine for adders whose width is not a power of 2 can be derived by removing unused logic from the spine of the next-larger power-of-2 adder.

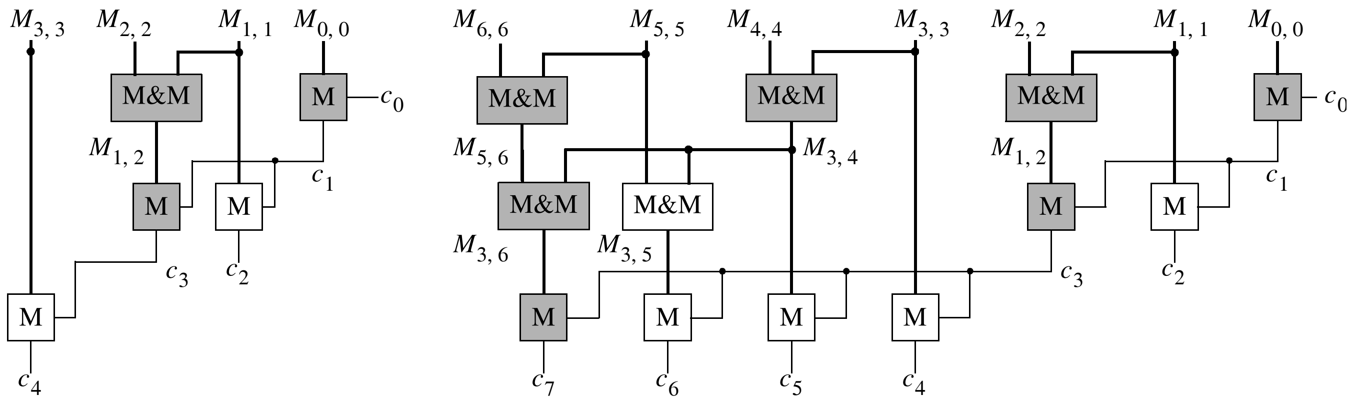


Figure 4-2: Minimum-depth and area 4- and 8-bit M&M CLA adder structure

The CLA structure for minimum-depth and minimum-area 4- and 8-bit M&M adders are shown Figure 4-2 where the spine cells are shaded. It is interesting to note that the adder synthesized by M31, shown in Figure 1-1, is not minimal in area.

5 G&P vs. M&M adders

To better understand the M&M decomposition of carry logic, we establish in this section its relation to the more traditional G&P decomposition. The basic relation is stated in the following theorem:

Theorem 5.1 For any $0 \le i \le k < n$ we have:

$$M^a_{i,k} M^b_{i,k} = G_{i,k}\text{, and} \tag{5.1}$$

$$M^a_{i,k} + M^b_{i,k} = G_{i,k} + P_{i,k} \tag{5.2}$$

Proof: We use induction to show the validity of (5.1); (5.2) can be proved similarly.

(Basis) For $k = i$, (5.1) is true by definition since $M^a_{i,i} M^b_{i,i} = a_i b_i = G_{i,i} \equiv g_i$.

(Induction) Suppose (5.1) holds for some k, i.e. $M^a_{i,k} M^b_{i,k} = G_{i,k}$. Then,

$$M^a_{i,k+1} M^b_{i,k+1}$$

$$= \mathrm{MAJ}(a_{k+1}, b_{k+1}, M^a_{i,k}) \cdot \mathrm{MAJ}(a_{k+1}, b_{k+1}, M^b_{i,k}) \qquad [M_{i,k}\text{ definition}]$$

$$= [a_{k+1} b_{k+1} + (a_{k+1} + b_{k+1}) M^a_{i,k}] \cdot [a_{k+1} b_{k+1} + (a_{k+1} + b_{k+1}) M^b_{i,k}] \qquad [\text{MAJ definition}]$$

Table 1: G&P vs. M&M CLA Logic

	G&P Structure		**M&M Structure**	
	$G_{i,k}$	$P_{i,k}$	$M^a_{i,k}$	$M^b_{i,k}$
$k = i$	$a_i \cdot b_i$	$a_i + b_i$	a_i	b_i
$k > j \geq i$	$G_{j+1,k} + P_{j+1,k} \cdot G_{i,j}$	$P_{j+1,k} \cdot P_{i,j}$	$\mathrm{MAJ}(M^a_{j+1,k}, M^b_{j+1,k}, M^a_{i,j})$	$\mathrm{MAJ}(M^a_{j+1,k}, M^b_{j+1,k}, M^b_{i,j})$
c_{k+1}	$G_{i,k} + P_{i,k} \cdot c_i$		$\mathrm{MAJ}(M^a_{i,k}, M^b_{i,k}, c_i)$	

$$= a_{k+1}b_{k+1} + (a_{k+1} + b_{k+1})M^a_{i,k}M^b_{i,k} \qquad \text{[Distributivity]}$$

$$= G_{k+1,k+1} + P_{k+1,k+1}M^a_{i,k}M^b_{i,k} \qquad [G_{i,i} \text{ and } P_{i,i} \text{ definition}]$$

$$= G_{k+1,k+1} + P_{k+1,k+1}G_{i,k} \qquad \text{[Induction hypothesis]}$$

$$= G_{i,k+1} \qquad [G_{i,k+1} \text{ definition}]$$

Thus, by induction, (5.1) holds for all $k \geq i$. ■

Table 1 shows a side-by-side comparison of the G&P and M&M decomposition equations for carry logic. We can make several observations about these two styles of CLA structure:

- The G&P structure requires an initial level of logic gates to produce the single-bit generate and propagate signals $G_{i,i}$ and $P_{i,i}$ from the addend and augend inputs. This is unnecessary for the M&M structure, enabling it to be implemented with one fewer levels of logic than an equivalent G&P structure.

- Beyond the first level, the G&P structure requires two types of gates: 2-input ANDs for the group propagate signals, and 3-input complex AO12 (AND-OR) gates for the group generate and carry signals. The M&M structure, in contrast, employs just one, albeit more complex, gate type, namely the 3-input majority.

- The structures described so far (G&P and M&M) are based on a group size of 2, yielding CLA structures whose minimum depth is $O(\log_2 n)$. G&P structures can be generalized for group sizes $m > 2$ that yield CLA structures whose minimum depth is $O(\log_m n)$ using AND and AND-OR gates with larger fan-in. M&M decomposition does not generalize in this manner because the distributive property of majority does not hold for larger (e.g. 5-input) majority gates. This restriction may not be significant in practice most VLSI implementations of CLA are based on a group size of 2.

Acknowledgments

We thank Ed Davidson for deriving the distributivity of majority.

6 References

[1] O.J. Bedrij, “Carry-select adder”, *IRE Trans. on Electron. Computers*, vol. EC-11, pp. 340-346, June 1962.

[2] R.P. Brent and H.T. Kung, “A regular layout for parallel adders”, *IEEE Transactions on Computers*, vol. EC-9, pp. 260-264, 1982.

[3] V. N. Kravets and K. A. Sakallah, “Constructive Library-Aware Synthesis Using Symmetries,” in *Proceedings of the Design, Automation and Test in Europe Conference (DATE 2000)*, pp. 208-213, Paris, France, 2000.

[4] Victor N. Kravets, “Constructive Multi-Level Synthesis by Way of Functional Properties,” *Ph.D. Thesis*, University of Michigan, May 2001.

[5] Richard E. Ladner and Richard E. Fischer, “Parallel Prefic Computation”, *Journal of the ACM (JACM)*, vol. 27, no 4, pp. 831-838, October 1980.

[6] M. Lehman and N. Burla, “Skip techniques for high-speed carry propagation in binary arithmetic units”, *IRE Trans. on Electron. Computers*, vol. EC-10, pp. 691-698, 1961.

[7] J. Sklansky, “Conditional-sum addition logic”, *IEEE Transactions on Electronic Computers*, vol. EC-9, pp. 226-231, June 1960.