Design of m-out-of-n Bit-Voters

Behrooz Parhami

Dept. of Electrical & Computer Engineering
University of California
Santa Barbara, CA 93106, USA

Abstract

Several synthesis methods for m-out-of-n bit-voting networks are described and the costs of the resultant designs are compared. In the special case of majority voting (with gate count used as the cost index), it is shown that a multiplexer-based design method yields the best realizations for small n, while designs based on selection networks are most efficient when n is large. Other design techniques are offered that might be more efficient with particular technologies or when certain standard building blocks are to be used in the hardware realization. Most of these designs can be extended to the general case of unequal vote weights (weighted voting).

0 Introduction

The idea of voting to obtain highly reliable results from multiple, less reliable, computation channels has been used extensively in the design of fault-tolerant computer systems. When voting is to be performed frequently and at a relatively low level, high-throughput hardware voters must be utilized. In this paper, we report some results from an ongoing research project dealing with general design paradigms for hardware bit- and word-voters as well as efficient algorithms for software voting on more complex data objects [2], [3], [4].

We consider the design of an m-out-of-n bit-voter (henceforth simply voter) defined as a circuit with n inputs x_i, $i = 1, 2, \ldots, n$, and a single output y, such that $y = 1$ iff at least m of the n inputs are 1. Note that our definition is asymmetric with respect to 0 and 1 in that an output of 0 does not imply and is not implied by the presence of at least m zeros at the inputs. Such asymmetry should not be worrisome and is in fact useful in the design of systems where one of the two output values is considered 'safe'. It is easy to modify the designs such that 0/1 symmetry is achieved and an extra output indicates when neither 0 nor 1 has the required quorum. Majority voting corresponds to the special case of $m = \lceil (n + 1)/2 \rceil$ and has 0/1 symmetry for odd n.

1 Gate-Level Designs

A voter can be constructed as a two-level AND-OR (equivalently NAND-NAND) digital logic circuit with $g = n!/[m!\ (n - m)!]$ m-input AND gates and a single g-input OR gate for small values of the parameters m and n. Also, a two-level OR-AND realization, requiring $g' = n!/[(m - 1)!\ (n - m + 1)!]$ $(n-m+1)$-input OR gates and a single g'-input AND gate, is possible. In the first realization, all distinct subsets of m inputs are ANDed together and the voter output is 1 if at least one of the AND results is 1. In the second realization, all possible subsets of $n - m + 1$ inputs are ORed together and the voter output is 0 if at least one of the OR results is 0.

The two-level AND-OR realization is 'simpler' than the two-level OR-AND version (in terms of both gate count and gate-input count) iff $m > (n + 1)/2$. The complexities are equal for odd n if $m = (n + 1)/2$. As an example, for a 2-out-of-5 voter, the two-level AND-OR design uses 10 two-input AND gates and a single 10-input OR gate while the OR-AND design is less complex with 5 four-input OR gates and one 5-input AND gate.

For large values of n, two-level designs are impractical. Assuming the use of f-input gates and ignoring the possibility of gate sharing, the total number of gates in the two-phase AND-OR and OR-AND realizations will change from $g + 1$ and $g' + 1$ to:

$$G = g\lceil (m - 1)/(f - 1) \rceil + \lceil (g - 1)/(f - 1) \rceil$$

$$G' = g'\lceil (n - m)/(f - 1) \rceil + \lceil (g' - 1)/(f - 1) \rceil$$

These expressions are obtained by replacing each gate with a tree of f-input gates implementing the same function. The circuit delays will increase to $\lceil \log_f m \rceil + \lceil \log_f g \rceil$ and $\lceil \log_f (n - m + 1) \rceil + \lceil \log_f g' \rceil$ gate levels.

With gate sharing, an exact general gate-count analysis becomes difficult. However bounds for the number of gates can be obtained that are close to actual values and show the excessive complexity of this approach for large values of n. It is thus imperative to explore more structured design techniques.

1058-6393/91 $1.00 © 1991 IEEE

2 Decomposition-Based Designs

Hierarchical decomposition strategy (divide-&-conquer) can be used to facilitate the design. There are two ways to proceed with the decomposition approach:

1. Picking a partitioning scheme and then designing a suitable merging network.
2. Selecting a merging network and then designing the required partitioning algorithm.

With the first approach, we divide the inputs into disjoint subsets, enumerate the various combinations in which different subsets can contribute votes in such a way that the voting threshold is matched or exceeded, provide smaller voters to realize these contributions, and finally, design a logic network for combining the results. Because the subsets can be selected in many different ways, this approach does not lend itself to general analyses. We will thus limit our discussion to a simple example.

Consider the design of a 3-out-of-5 voter using the subsets $S_1 = \{x_1, x_2, x_3\}$ and $S_2 = \{x_4, x_5\}$. The combinations that match or exceed the threshold of 3 are:

3-of-3 in S_1 + (2-of-3 in S_1 and 1-of-2 in S_2) + (1-of-3 in S_1 and 2-of-2 in S_2)

This yields the logical expression $x_1x_2x_3 + (x_1x_2 + x_2x_3 + x_3x_1)(x_4 + x_5) + (x_1 + x_2 + x_3)x_4x_5$ which directly translates into a 4-level logic circuit with 10 gates and 25 input lines.

We next explore the second decomposition strategy with multiplexers used as merging networks. Our interest in this approach arises from the availability of multiplexers as off-the-shelf universal components. The strategy is to select a subset of the inputs as control inputs to a multiplexer, determine the residual input functions, and then repeat the process for each function, if needed, until easily realizable functions are obtained.

For example, with a 2-input multiplexer in the first decomposition stage, the residual functions correspond to an m-out-of-$(n-1)$ voter and an $(m-1)$-out-of-$(n-1)$ voter. To design a 3-out-of-5 voter using 2-input multiplexers, we take x_1 as the first control variable. The residual functions corresponding to $x_1 = 0$ and $x_1 = 1$ are $x_2x_3x_4 + x_2x_3x_5 + x_2x_4x_5 + x_3x_4x_5$ and $x_2x_3 + x_2x_4 + x_2x_5 + x_3x_4 + x_3x_5 + x_4x_5$, yielding the result

$$[x'_1[x'_2(x_3x_4x_5)+x_2h] + x_1[x'_2h+x_2(x_3+x_4+x_5)]]$$

where $h = x_3x_4 + x_3x_5 + x_4x_5$ has the 2-input multiplexer realization $h = [x'_3(x_4x_5) + x_3(x_4+x_5)]$. The resulting circuit is shown in Figure 1a. Clearly, a 4-input multiplexer can replace the last two levels. With 8-input multiplexers, the expression becomes

$$[x'_1x'_2x_3h_1 + x'_1x_2x'_3h_1 + x'_1x_2x_3h_2 + x_1x'_2x'_3h_1 + x_1x'_2x_3h_2 + x_1x_2x'_3h_2 + x_1x_2x_3]$$

where $h_1 = (x_4x_5)$ and $h_2 = (x_4 + x_5)$. The resulting circuit is depicted in Figure 1b.

3 Arithmetic-Based Design

In the 'arithmetic' approach, the sign of $-m + \sum x_i$ is computed and the voting result is the complement of this sign. A parallel counter [5] to compute $\sum x_i$ followed by a circuit to compare the sum to m (or a single 'accumulative parallel counter') can easily compute the output. Designs for parallel counters are based on carry-save reduction techniques which are used extensively in reducing multiple-operand additions to two-operand additions. Cost and delay analyses for such reductions are available [5], [6]. One advantage of the arithmetic-based design is that it can be easily extended to the case of unequal vote weight as discussed in Section 6.

4 Design with Selection Networks

The design of an m-out-of-n voter is equivalent to selecting the mth largest value from among n input bits. Selection networks can be built from 2-sorter (comparator) cells. Knuth [1] defines three types of selection networks with n inputs:

1. Select the m largest values and move them to m outputs in no particular order.
2. Select the mth largest value and move it to a specified output line.
3. Select the m largest values and move them to m output lines in sorted order.

Denoting the number of 2-sorter or comparator cells by $U(m, n)$, $V(m, n)$, and $W(m, n)$ for type-1, type-2, and type-3 selectors above, we have:

$$U(m, n) \leq V(m, n) \leq W(m, n)$$

When dealing with bits, a two-sorter simply consists of a pair of 2-input gates: An OR to produce the larger and an AND to produce the smaller of the two values.

Type-3 selectors do more than what is required here. Type-2 selectors do exactly what we want. However, for most practical values of m and n, a type-1 selector augmented by an AND or OR circuit (that indicates whether all of the m largest values are 1s or whether all of the $n - m + 1$ smallest values are not all 0s) is both faster and more economical.

Consider the design of a 4-out-of-8 voter. The required type-1 selection network that selects the 4 largest bit values and moves them to the upper half of the output lines is given in Figure 2, where each heavy vertical line represents a comparator that moves the larger of the two input values to the upper and the smaller value to the lower line. This selector requires 14 comparators or 28 two-input gates with 4 gate levels of delay. A 4-input

AND gate connected to the upper 4 outputs completes the circuit. Note that a 5-out-of-8 majority voter results if we connect an OR gate to the lower 4 output lines.

5 Comparison of Various Designs

We will compare the designs only for simple majority voters (i.e., when $m = \lfloor n/2 \rfloor + 1$). Figure 3 shows the cost of majority voters designed based on 2-level logic expressions ('gate-level'), two-input multiplexer decomposition, the arithmetic-based approach, and selection networks, assuming maximum gate fan-in of 4. Figure 3 indicates that the gate-level or multiplexer-based approach is best for small values of n whereas selection networks offer the most economical solution for larger values of n. The theory of selection networks is well-developed and efficient designs are available [1].

A comparison of delays is much more difficult. If the designs are used with pipelining [2], the differences in latencies (number of gate levels) are not significant as far as throughput is concerned. However, the number of gate levels does affect the cost due to the requirement for latches between pipeline stages. A general analysis is impractical because the number of logic signals going from one pipeline stage to the next cannot be expressed as a simple function of the relevant parameters.

6 Weighted Bit-Voters

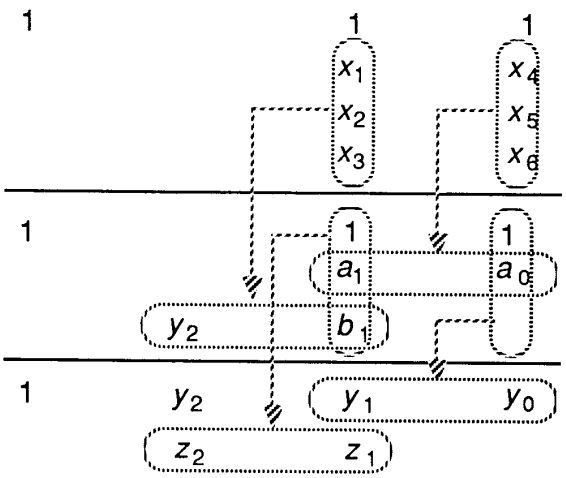
If each of the n bit-inputs x_i, $i = 1, 2, \ldots, n$, has an associated vote v_i such that the output 1 is produced iff the sum of votes associated with the inputs having the value 1 is no less than a threshold t, then we have a generalized t-out-of-Σv_i voter.

The techniques discussed in the preceding sections can be extended for such voters. The only exception concerns selection networks that can be used here only if the vote weights are small integers. The way to do this is to fan-out each input to v_i inputs of a (Σv_i)-input voter. However such designs tend to be highly inefficient.

The arithmetic-based approach is ideally suited to this case. As a simple example, consider the design of 6-input voter where the inputs have the associated votes of 2, 2, 2, 1, 1, 1 and the voting threshold is 5. The arithmetic expression to be evaluated is

$$-5 + 2x_1 + 2x_2 + 2x_3 + x_4 + x_5 + x_6$$

The reduction steps in our multiple-operand addition using full-adder and half-adder cells are shown in the following diagram. Here, 1011 is the 4-bit 2's-complement representation of the constant –5. Three of the inputs are written in the second column because they carry a weight of 2. If the weight of x_1 were 3, say, then x_1 would be written in both the first and the second column. Each vertical box encloses the inputs to a full or half adder and the associated horizontal box encloses the two outputs. The diagram shows level 1 with 2 full adders and level 2 with 1 full adder and 1 half adder. A 2-bit adder can be used to reduce the remaining y and z bits in the middle two columns, with the carry-out of this adder providing the final output. The leftmost 1 can be ignored since it causes only a complementation that cancels the complementation needed for obtaining the resultant output from the sign bit.



A particularly useful arithmetic-based design scheme is depicted in Figure 4. Here a pair of binary trees, composed of adders of appropriate lengths, tally the votes for 0 and 1 inputs. The 'compare and select' circuit at the right produces the final output. The number of adders required for an n-input voter is $2n - 2$ and the longest adder deals with operands of length $\log_2(\Sigma v_i)$.

The multiplexer-based approach works fine with weighted voting if proper care is taken to select the control variables in an optimal order.

7 Conclusions

A theoretical study of weighted voting [4] has shown that voting is at least as complex as sorting when the size of the input space is large. On the other hand with a small input space of size δ ($\delta = 2$ for bit-voting), linear-size, logarithmic-time voters are practically realizable. In this paper, we have explored and compared some useful design techniques for m-out-of-n and the more general t-out-of-Σv_i weighted bit-voters.

Despite the fact that the designs are quite practical, and in some cases asymptotically optimal, no claim is made as to their absolute efficiency or optimality. There may be other methods that yield better designs for a given set

of requirements. The optimality question and several extensions to the design techniques discussed here are being investigated as part of an ongoing research project.

References

[1] Knuth, D.E., *The Art of Computer Programming — Vol. 3: Sorting and Searching*, Addison-Wesley, 1973, Section 5.3.4, pp. 220-246.

[2] Parhami, B., "High-Performance Parallel Pipelined Voting Networks", *Proc. of the International Parallel Processing Symp.*, Anaheim, CA, Apr./May 1991, pp. 491-494.

[3] Parhami, B., "Voting Networks", *IEEE Transactions on Reliability*, Vol. 40, pp. 380-394, Aug. 1991.

[4] Parhami, B., "The Parallel Complexity of Weighted Voting", *Proc. of the International Conf. on Parallel and Distributed Computing and Systems*, Washington, DC, Oct. 1991, pp. 382-385.

[5] Swartzlander, E.E., "Parallel Counters," *IEEE Transactions on Computers*, Vol. C-22, No. 11, Nov. 1973, pp. 1021-1024.

[6] Waser, S. and M.J. Flynn, *Introduction to Arithmetic for Digital System Designers*, Holt, Rinehart, & Winston, 1982.

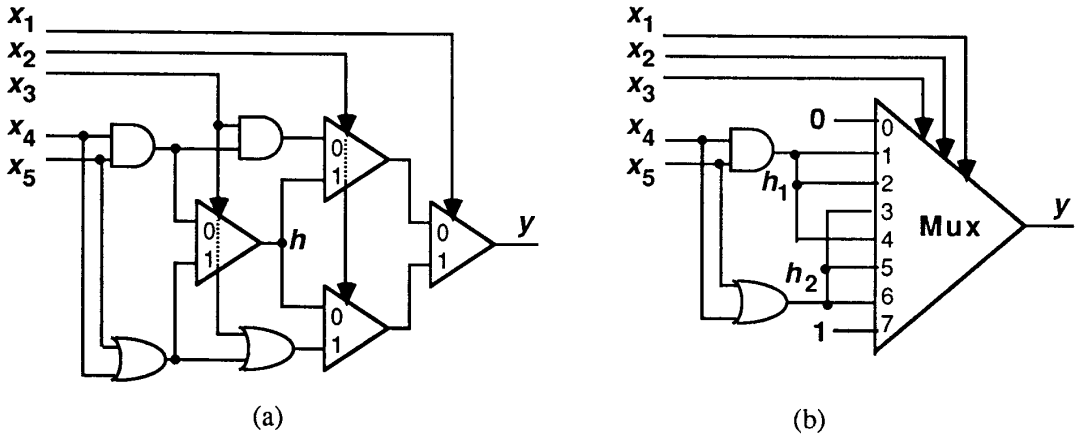


Figure 1. Multiplexer-based realizations for a 3-out-of-5 bit-voter.

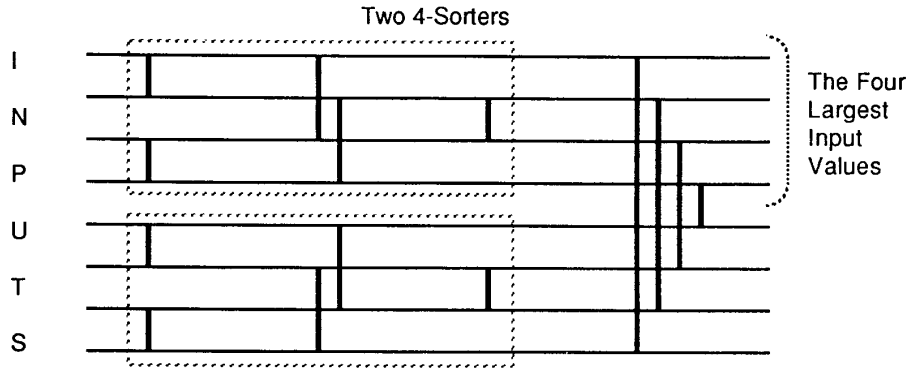


Figure 2. Network of comparators to select the 4 largest of 8 values.

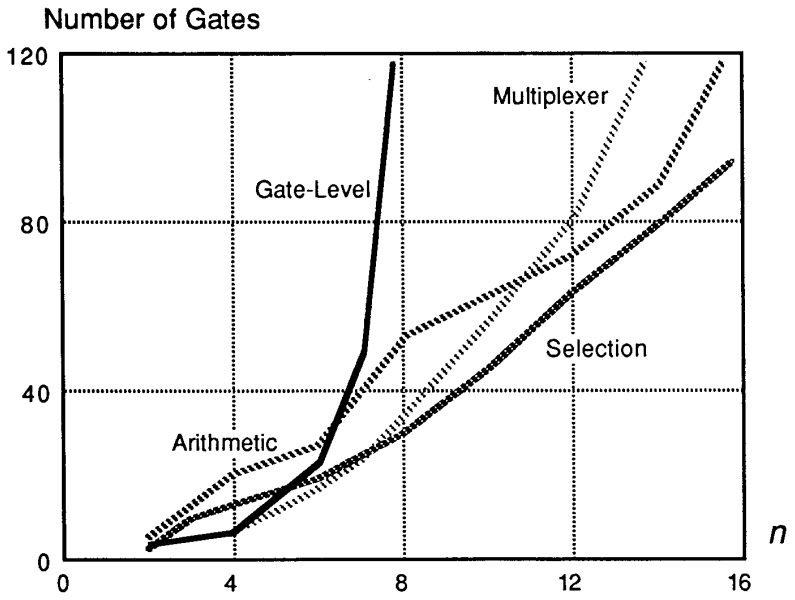


Figure 3. Cost of simple majority bit-voters with different designs as a function of input size n.

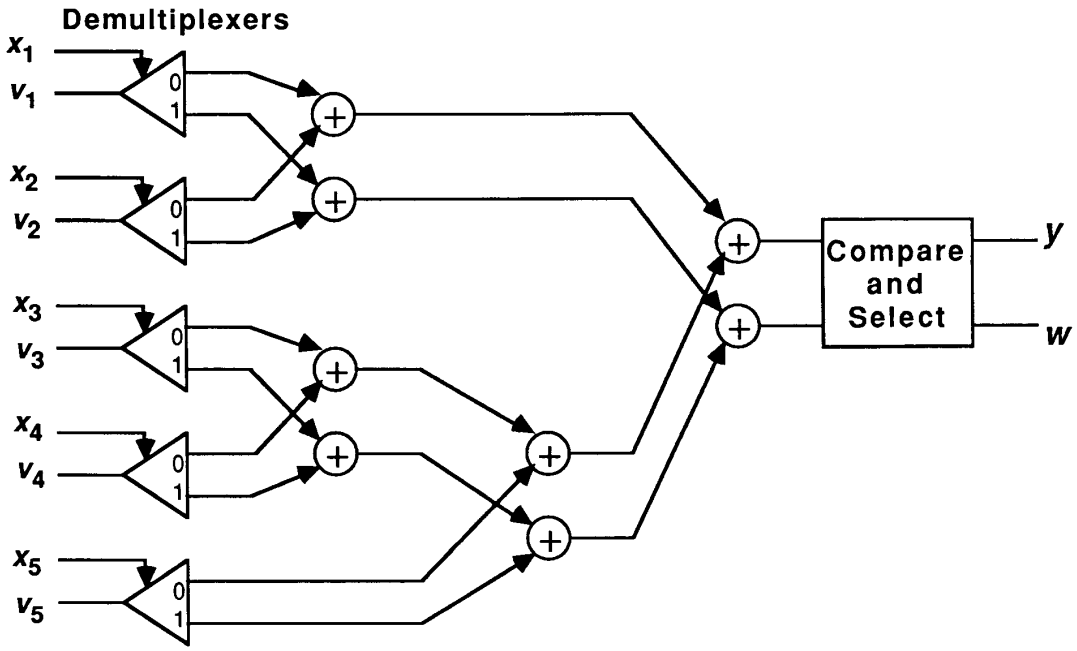


Figure 4. A 5-input weighted bit-voter constructed according to the 'vote-tallying' method.