PARALLEL COUNTERS FOR SIGNED BINARY SIGNALS

Behrooz Parhami

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

ABSTRACT

A parallel counter is a combinational logic circuit that receives a set of binary count signals in parallel and determines the final count after some fixed delay. In this paper, a more general parallel counter is presented whose count inputs have three states (i.e. down, none, and up or, equivalently, –1, 0, and 1). Such parallel up/down counters find applications in arithmetic circuits dealing with binary signed-digit numbers, in parallel search schemes where one must combine "approve," "disapprove" and "abstain" responses, and in certain signal processing problems.

1. BACKGROUND

In its simplest form, a *counter* is a sequential circuit that stores an integer value and can increment (and, in the case of up/down counters, also decrement) it by 1 upon the receipt of a special "enable" or "count" signal. Although techniques are available for designing high-speed counters with conventional number representations [OBER81], the speeds that can be achieved are limited by the requirement for carry propagation. An alternative is to take advantage of the carry-free addition property of redundant number representations [AVIZ61], [METZ59], [PARH87], [PARH87a], [PARH89], but even with this method, speeds will be limited by the basic switching characteristics of components and by the need for final conversion of the redundant count into a conventional binary number. Thus, to count the number of 1s among many thousands of bits by feeding them sequentially to a high-speed counter of either type would still imply delays well in excess of several tens of microseconds and such a delay may be unacceptable in certain applications. Clearly, some form of parallelism is needed to achieve higher speeds.

A *parallel counter* [SWAR73], or an (n, m) counter, has been defined as a combinational digital logic circuit having n binary inputs and $m = \lfloor \log_2 n \rfloor + 1$ binary outputs (or alternatively, having between $n = 2^{m-1}$ and $n = 2^m - 1$ binary inputs for m binary outputs), where the outputs correspond to the base-2 representation of the sum of the n input bits; i.e., a number between 0 and n. Such parallel "up" counters have been studied in connection with counting of multiple responders in associative memories and processors [FOST71] and finding the sum of a column of 1s in multi-operand adders and in high-speed parallel array multipliers [SING73], [DADD65], [SVOB70], with numerous implementation alternatives investigated [CURR79], [CURR80], [DADD80], [DORM81], [DORM82], [GAJS80], [KOBA78], [LAIH82], [MEOA75].

Up/down sequential counters can count down as well as up. This property is normally implemented by providing separate "up-count" and "down-count" binary signals or by a pair of "count" and "mode" signals as inputs to the counter. Such counters can be viewed as receiving a binary-encoded three-state count signal denoting an "increment" in the set {–1, 0, 1} at each cycle and adding this value to the current stored count. The parallel version of such a counter adds a finite set of signals from the set {–1, 0, 1}, although a more appropriate generalization would require adding the sum of such a set of "signed binary" signals to an accumulated count. In this paper, we ignore this "accumulative" property and concentrate instead on the design of parallel up/down counters as purely combinational circuits.

Just as parallel "up" counters are useful for deterrmining the multiplicity of responders to a parallel search-type operation in an associative memory [FOST71] where the responses are binary ("yes/no") in nature, parallel "up/down" counters find applications in similar situations where three-valued or ternary ("agree/disagree/no-opinion") responses must quickly be combined. One way to handle such ternary responses is to decompose them into two binary responses, use two parallel up counters (or a single such counter twice) to tally each response separately, and finally subtract the two output values to find the actual count. However, speed can be gained by utilizing a special hardware unit to obtain the count directly.

2. ENCODING SCHEMES

To implement an up/down parallel counter, two encoding issues must be resolved:

1. Encoding of the output integer count value.
2. Encoding of the three-state input count signals.

Although three-valued logic [SMIT88] finds a natural application here and does in fact provide a viable design alternative, we will assume the use of bianry encodings and standard logic in all of our implementations.

We will consider two encodings for the output count value: standard two's complement and binary signed-digit. The first encoding is desirable because the counter output will be usable directly in other parts of the computations and the second encoding offers the advantage of design simplicity (as we will see shortly) but may necessitate a conversion from binary signed-digit to standard two's-complement format.

23ACSSC-12/89/0513 $1.00

As for the three-state count signals, there are 96 possible encodings for 3 values with 2 bits. Many of these assignments can be eliminated by ignoring permutations and negations. Robertson's list of 9 distinct assignments [CHOW78], with three of the assignments modified here by complementing the left bit in order to make 00 represent 0 in all cases, is given in Table I.

Table I

The 9 distinct assignments of 2-bit code vectors to digits in the set {−1, 0, 1}.

	Code Vectors				
Assign.	00	01	10	11	Comments on encoding
R2	0	1	x	−1	$\langle s, v\rangle$
R5'	0	−1	x	1	$\langle s, v\rangle$ with reverse sign
R4	0	1	−1	x	$\langle n, p\rangle$
R3	0	1	0	−1	$\langle s, v\rangle$ with both 0 and −0
R1	0	1	−1	0	$\langle n, p\rangle$ with 11 denoting 0
R6	0	1	1	−1	
R8'	0	−1	1	1	$\langle p, n\rangle$ with p dominating
R7	0	1	−1	−1	$\langle n, p\rangle$ with n dominating
R9'	0	−1	−1	1	

Assignment numbers in the first column of Table I indicate the numbering of Robertson's assignments in the cited reference and primed numbers correspond to modified encodings as explained above. The $\langle s, v\rangle$ encoding is the natural sign-and-value encoding with $s = 1$ representing a negative sign and the $\langle n, p\rangle$ or $\langle p, n\rangle$ encoding consists of "negative" and "positive" flags, one of which is set to denote a nonzero digit ($p=1$ representing 1 and $n=1$ representing −1). These can also be interpreted as two's-complement and one's complement two-bit encodings.

With two's-complement output, the cases with don't-cares (i.e. R2, R5', and R4 in Table I) will result in simpler realizations (thus only the $\langle s, v\rangle$ and $\langle n, p\rangle$ encodings need to be considered) whereas with binary signed-digit output, the encodings with double codes for one of the values must also be dealt with. In the latter case, the flexibility gained in the generation of one of two possible output codes may offset the loss of don't-care conditions in the input. Despite the above observation, in this paper we will limit our discussion to simple $\langle s, v\rangle$ and $\langle n, p\rangle$ encodings in all cases above.

If binary signed-digit representation is chosen for the final count, then an obvious way to design a parallel up/down counter is to realize a "signed full adder" cell for three signed binary digits such that the two output signed bits represent the radix-2 sum of the three signed input bits (note that this signed full adder is different from a binary signed-digit adder cell [CHOW78]). Once such a signed full adder cell is available, it can replace the standard full adder cell in any parallel up counter design to obtain a parallel up/down counter. Thus, we first consider the design of a signed full adder cell.

3. SIGNED FULL ADDERS

A signed full adder (SFA) receives three signed binary digits as input and produces two signed binary digits (sum and carry) as output. In this section, we present designs for such full adders with the $\langle s, v\rangle$ and $\langle n, p\rangle$ encodings and will compare them with standard full adders. In both cases, subscripts 1, 2, and 3 will identify the three input values while "double-primed" and "primed" names designate carry and sum digits, respectively.

Five of the seven possible output values of a signed full adder (i.e. −3, −2, 0, +2, +3) have unique representations with binary signed digits, while −1 and +1 each can be represented by two carry-sum pairs: (0, −1) or (−1, 1) and (0, 1) or (1, −1). This results in "coupled" don't-care states in the realization of the carry and sum outputs and renders the logic minimization problem more difficult.

With the $\langle s, v\rangle$ encoding, minimal two-level sum-of-products realization of a signed full adder is as follows (the notation is explained above):

$$s'' = s_1 s_2 s_3 + s_1 s_2 v'_3 + s_2 s_3 v'_1 + s_3 s_1 v'_2$$

$$v'' = s'' + s'_1 s'_2 s'_3 v_1 v_2 + s'_1 s'_2 s'_3 v_2 v_3 + s'_1 s'_2 s'_3 v_3 v_1$$

$$s' = s_1 s_2 v_3 + s_2 s_3 v_1 + s_3 s_1 v_2 + s_1 v'_2 v'_3 + s_2 v'_3 v'_1 + s_3 v'_1 v'_2$$

$$v' = v_1 v_2 v_3 + v_1 v'_2 v'_3 + v'_1 v_2 v'_3 + v'_1 v'_2 v_3$$

These expressions correspond to the case where the sums of −1 and 1 are output as the pairs (0, −1) and (0, 1), respectively. The other three combinations result in more complex realizations.

With the $\langle n, p\rangle$ encoding, minimal two-level sum-of-products realization of a signed full adder is as follows:

$$n'' = n_1 n_2 + n_2 n_3 + n_3 n_1$$

$$p'' = p_1 p_2 + p_2 p_3 + p_3 p_1$$

$$n' = n_1 n_2 p_3 + n_2 n_3 p_1 + n_3 n_1 p_2 + n_1 p'_2 p'_3 + n_2 p'_3 p'_1 + n_3 p'_1 p'_2$$

$$p' = p_1 p_2 n_3 + p_2 p_3 n_1 + p_3 p_1 n_2 + p_1 n'_2 n'_3 + p_2 n'_3 n'_1 + p_3 n'_1 n'_2$$

Again, these expressions correspond to the case where the sums of −1 and 1 are output as the pairs (0, −1) and (0, 1), respectively.

Table II

Complexity of signed full adders compared to conventional full adders.

Complexity Parameter	$\langle s,v\rangle$-SFA	$\langle n,p\rangle$-SFA	FA
I/O connections	10	10	5
No. of logic gates	21	22	9
No. of gate inputs	78	66	25

Table II shows a comparison of complexities for $\langle s, v\rangle$-encoded and $\langle n, p\rangle$-encoded SFAs to that of conventional full adders. The speeds (delays) are roughly the same as two-level logic is used in both cases with small variations on average gate fan-in. As seen from Table II, the number of I/O connections is doubled, while the number of logic gates and gate inputs are multiplied by 2.33 and 3.12, respectively, for $\langle s, v\rangle$-encoded SFAs and by 2.44 and 2.64 for $\langle n, p\rangle$-encoded SFAs. The $\langle n,p\rangle$ encoding is thus more efficient; a conclusion which is consistent which results for other types of arithmetic circuits on binary signed-digit numbers where the $\langle n, p\rangle$ encoding offers similar advantages over the $\langle s, v\rangle$ encoding [PARH88].

4. COST AND DELAY

As stated earlier, a parallel up/down counter with binary signed-digit representation of the final count is obtained if we replace each full-adder cell in a parallel up counter with a signed full adder. Therefore, all of the developed theories for parallel up counters (including the cost and delay analyses) carry over directly if we replace the cost and delay of a standard full adder with the corresponding parameters of a signed full adder. The delay of an SFA is the same as that of an FA. The cost parameters are given in Table II.

An n-input parallel up counter requires approximately n standard FA cells (more precisely, at least $n - m$ and no more than n FAs) organized into a tree-like structure of depth (delay) between

$$\delta_{min}(n) = \lceil \log_3 n \rceil + \lfloor \log_2 n \rfloor - 1$$

and $\delta_{max}(n)$ obtained from the following recurrence

$$\delta_{max}(n) = \delta_{max}(\lfloor n/2 \rfloor) + 2$$

with $\delta_{max}(3) = \delta_{max}(2) = 1$, where the unit of delay is that of a full adder; i.e. 2 logic levels. The expression for $\delta_{min}(n)$ is derived by observing that at least $\lceil \log_3 n \rceil$ levels of full adders are required for computing the least significant bit of the count and $\lfloor \log_2 n \rfloor - 1$ stages are present in the final carry propagation path. The recurrence for $\delta_{max}(n)$ is based on the observation that an n-input parallel counter can be synthesized from two $(\lfloor n/2 \rfloor)$-input parallel counters and a $(\lfloor \log_2(\lfloor n/2 \rfloor) \rfloor + 1)$-bit ripple-carry adder. Each ripple-carry adder in the resulting tree structure is one bit longer than its predecessors and contributes an additional two stages of delay. In practice, the actual delay is usually closer to $\delta_{max}(n)$ than to $\delta_{min}(n)$ [FOST76].

Rather than adapting parallel up counter designs, one can use the limited-carry property of binary signed-digit numbers to create a different type of design. The new design is also based on the divide-and-conquer strategy. An n-input parallel up/down counter is synthesized from two $(\lfloor n/2 \rfloor)$-input counters plus a $(\lfloor \log_2(\lfloor n/2 \rfloor) \rfloor + 1)$-bit binary signed-digit adder. Then, the delay recurrence becomes

$$\delta_{max}(n) = \delta_{max}(\lfloor n/2 \rfloor) + 1$$

where the unit of delay is now the time required for a limited-carry binary signed-digit addition. It has been shown [PARH88] that this addition process can be performed by a recoding stage requiring 2 logic levels followed by a carry-free addition stage realized by another 2 logic levels. Thus, delays of the two schemes are comparable.

Assuming $\langle n, p\rangle$-encoded signals, the total complexity (recoding plus addition) for each binary signed-digit adder stage is 23 logic gates having 75 input lines [PARH88]. Although this is more complex than the signed binary adder (Table II), the new scheme can be easily pipelined to achieve twice the throughput because the recoding stage and the subsequent carry-free addition stage are essentially independent. This increased throughput justifies the added complexity in some applications.

Parallel up/down counters with two's-complement output can be designed in a similar way. If $\langle s, v\rangle$-encoded inputs are used, the adder tree structure of parallel up counters is applicable with minor modifications to account for the signed values. With $\langle n,p\rangle$-encoded input signals, either a first level of special adders must be used to convert groups of three $\langle n, p\rangle$-encoded signed signals into 3-bit two's-complement numbers and then continuing as before or else a single level of OR gates must be utilized at the input to translate each $\langle n, p\rangle$-encoded signal to the equivalent $\langle s, v\rangle = \langle n, n + p\rangle$ pair. In either case, the cost and delays are roughly the same.

5. CONCLUSION

We have investigated design and implementation issues for parallel up/down counters and presented cost and delay analyses for various design alternatives, comparing each to the respective parameters of parallel up counters. The conclusion is that handling of signed signals is only slightly more complicated than that of unsigned signals in standard parallel counters. The author is currently pursuing several ideas for generalizing the designs to accumulative and modular parallel counters.

In addition to the two-bit encodings of signed binary signals considered here, it is also possible to use 1-out-of-3 encoding of the binary count signals to provide added flexibility and/or fault tolerance. The resulting $\langle n, o, p\rangle$ code consists of "negative," "zero," and "positive" flags. Such a "three-rail" implementation, which has also been found useful in the design of high-speed multipliers using binary signed-digit numbers [TAKA87], offers strong error detection properties at the expense of a larger number of interconnections.

REFERENCES

[AVIZ61] Avizienis, A., "Signed-Digit Number Representation for Fast Parallel Arithmetic," *IRE Transactions on Electronic Computers*, Vol. EC-10, pp. 389-400, 1961.

[CHOW78] Chow, C.Y. and J.E. Robertson, "Logical Design of a Redundant Binary Adder," *Proc. of the International Symp. on Computer Arithmetic*, Santa Monica, CA, Oct. 1978, pp. 109-115.

[CURR79] Current, K.W. and D.A. Mow, "Implementing Parallel Counters with Four-Valued Threshold Logic," *IEEE Transactions on Computers*, Vol. C-28, No. 3, pp. 200-204, Mar. 1979.

[CURR80] Current, K.W., "Pipelined Binary Parallel Counters Employing Latched Quaternary Logic Full Adders," *IEEE Transactions on Computers*, Vol. C-29, No. 5, pp. 400-403, May 1980.

[DADD65] Dadda, L., "Some Schemes for Parallel Multipliers," *Alta Frequenza*, Vol. 34, pp. 349-356, Mar. 1965.

[DADD80] Dadda, L., "Composite Parallel Counters," *IEEE Transactions on Computers*, Vol. C-29, No. 10, pp. 940-946, Oct. 1980.

[DORM81] Dormido, S. and M.A. Canto, "Synthesis of Generalized Parallel Counters," *IEEE Transactions on Computers*, Vol. C-30, No. 9, pp. 699-703, Sep. 1981.

[DORM82] Dormido, S. and M.A. Canto, "An Upper Bound for The Synthesis of Generalized Parallel Counters," *IEEE Transactions on Computers*, Vol. C-31, No. 8, pp. 802-805, Aug. 1982.

[FOST71] Foster, C.C. and F.D. Stockton, "Counting Responders in an Associative Memory," *IEEE Transactions on Computers*, Vol. C-20, No. 12, pp. 1580-1583, Dec. 1971.

[FOST76] Foster, C.C., *Content-Addressable Parallel Processors*, Van Nostrand Reinhold, New York, 1976, p. 166.

[GAJS80] Gajski, D.D., "Parallel Compressors," *IEEE Transactions on Computers*, Vol. C-29, No. 5, May 1980.

[KOBA78] Kobayashi, H. and H. O'Hara, "A Synthesizing Method for Large Parallel Counters with a Network of Smaller Ones," *IEEE Transactions on Computers*, Vol. C-27, No. 8, pp. 753-757, Aug. 1978.

[LAIH82] Lai, H.C. and S. Muroga, "Logic Networks of Carry-Save Adders," *IEEE Transactions on Computers*, Vol. C-31, No. 9, pp. 870-882, Sep. 1982.

[MEOA75] Meo, A.R., "Arithmetic Networks and Their Minimization Using a New Line of Elementary Units," *IEEE Transactions on Computers*, Vol. C-24, No. 3, pp. 258-280, Mar. 1975.

[METZ59] Metze, G. and J.E. Robertson, "Elimination of Carry Propagation in Digital Computers," *Information Processing '59* (Proc. of the UNESCO Conf.), June 1959, pp. 389-396.

[OBER81] Oberman, R.M.M., *Counting and Counters*, MacMillan, London, 1981.

[PARH87] Parhami, B., "Systolic Up/Down Counters with Zero and Sign Detection," *Proc. of the Eighth International Symp. on Computer Arithmetic*, Como, Italy, May 1987, pp. 174-178.

[PARH87a] Parhami, B., "A General Theory of Carry-Free and Limited-Carry Computer Arithmetic," Proc. of the Canadian Conf. on VLSI, Winnipeg, Oct. 1987, pp. 167-172.

[PARH88] Parhami, B., "Carry-Free Addition of Recoded Binary Signed-Digit Numbers," *IEEE Transactions on Computers*, Vol. 37, No. 11, pp. 1470-1476, Nov. 1988.

[PARH89] Parhami, B., "Generalized Signed-Digit Number Systems: A Unifying Framework for Redundant Number Representations," *IEEE Transactions on Computers*, to appear late 1988 or early 1990.

[SING73] Singh, S. and R. Waxman, "Multiple Operand Addition and Multiplication," *IEEE Transactions on Computers*, Vol. C-22, No. 2, pp. 113-119, Feb. 1973.

[SMIT88] Smith, K.C., "Multiple-Valued Logic: A Tutorial and Appreciation," *Computer*, Vol. 21, No. 4, Apr. 1988, pp. 17-27.

[SVOB70] Svoboda, A., "Adder with Distributed Control," *IEEE Transactions on Computers*, Vol. C--19, No. 8, pp. 749-751, Aug. 1970.

[SWAR73] Swartzlander, E.E., Jr., "Parallel Counters," *IEEE Transactions on Computers*, Vol. C-22, No. 11, pp. 1021-1024, Nov. 1973.

[TAKA87] Takagi, N. and S. Yajima, "On-Line Error-Detectable High-Speed VLSI Multiplier Using Redundant Binary Representation and Three-Rail Logic," *IEEE Transactions on Computers*, Vol. 36, No. 11, pp. 789-796, Nov. 1987.