On Using 1-out-of-n Codes for (p,q) Counter Implementations

Robert McIlhenny
rmcilhen@cs.ucla.edu

Miloš D. Ercegovac
milos@cs.ucla.edu

Computer Science Department
University of California
Los Angeles, CA 90024

Abstract

A new approach for implementing (p,q) counters is introduced, using 1-out-of-n code modules. The circuits were implemented in 1.2μM CMOS technology, and simulated using HSpice to measure cost, delay, and average power consumption. Through simulation, the new method is shown to yield an average 19% reduction in critical delay, and an average 30% reduction in average power consumption, with the tradeoff of a 38% increase in average cost.

1. Introduction

Past research in digital logic design has been directed toward increasing the speed of circuits. The recent demand for portable systems and moderate improvements in battery improvements have placed a demand on reducing power consumption, making it also a major critical design parameter [2]. Past metrics of performance include cost and delay. However the recent demand for low-power has shifted the attention toward metrics of performance defined by cost, delay, and power consumption.

Dynamic power consumption is directly proportional to the *switching activity*– the fraction of input transitions in which a 0 to 1 transition occurs at some output node in a circuit. The switching activity P_i of node i can be computed as:

$$P_i(0 \rightarrow 1) = P_i(0)P_i(1) = P_i(0)[1 - P_i(0)] \quad (1)$$

In this paper, we introduce a new approach of circuit implementation for parallel counters. This approach utilizes 1-out-of-n code modules and encoders. We compare the new approach to the conventional approach to demonstrate a significant reduction in both critical delay and average power comsumption, with the tradeoff of a generally larger cost.

A parallel p-to-q counter, denoted (p, q) is a logic device with p binary inputs $\underline{x} = (x_{p-1}, x_{p-2}, \ldots, x_1, x_0)$ and q outputs $\underline{z} = (z_{q-1}, z_{q-2}, \ldots, z_1, z_0)$, where $q = \lceil \log_2 p \rceil$ and z is the binary representation of the number of 1's in x [4]. It is also referred to in literature as a p-bit population counter [13] and a counting responder [6], and as a (p, q) adder. Parallel counters are used extensively in parallel multipliers [4], multiple-input adders [12], and various arithmetic circuits. More recently, they have been applied toward the design of digital neural networks [14], and triggering in multichannel high-energy spectrometers [11]. Improvements in design at the counter level have a significant impact on the performance of larger arithmetic circuits. We review the conventional approach and present our new approach for counter implementation.

2. Conventional Approach

The conventional approach for the implementation of parallel counters consists of a tree of half-adders and full-adders [4] [7], also referred to as (2,2) and (3,2) counters, respectively. The schematics of these circuits, denoted as HA1 and FA1, respectively, in [8] are shown in Figure 1. The schematics for a (4,3), (5,3), (6,3), and (7,3) counter are shown in Figure 2.

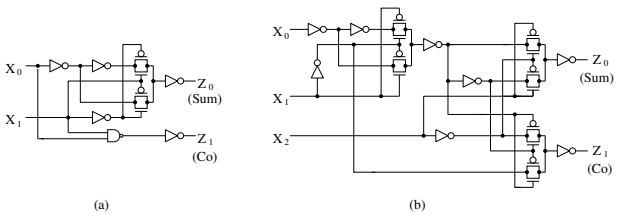

Figure 1. (a) HA1 and (b) FA1 schematics

The total cost of a (p,q) counter, based on this approach is $(p - q)$ full-adders, with the number of half-adders in the

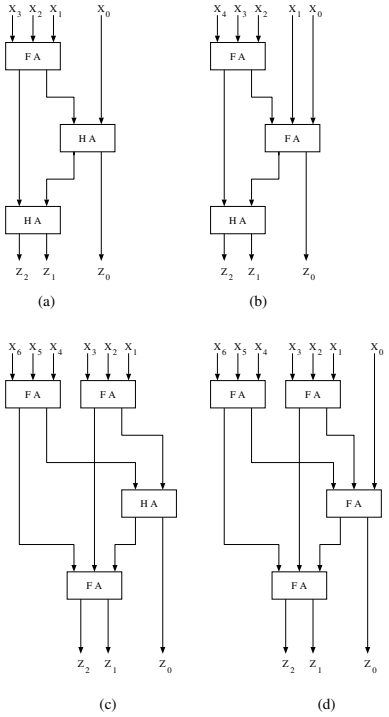

Figure 2. (a)(4,3), (b)(5,3), (c)(6,3), (d)(7,3) counter

range $\{0..q\}$ as shown in [3]. An upper bound on the delay, assuming a unit delay full-adder, as shown in [12] is

$$\delta = 2\lfloor log_2 p \rfloor - 1 \tag{2}$$

The switching activity of a half-adder's sum bit has and carry-out bit $haco$, and a full-adder's sum bit fas and carry-out bit $faco$ are shown below:

$$P_{has} = P_{has}(0)P_{has}(1) = (\frac{1}{2})(\frac{1}{2}) = \frac{1}{4} \tag{3}$$

$$P_{haco} = P_{haco}(0)P_{co}(1) = (\frac{3}{4})(\frac{1}{4}) = \frac{3}{16} \tag{4}$$

$$P_{fas} = P_{fas}(0)P_{fas}(1) = (\frac{1}{2})(\frac{1}{2}) = \frac{1}{4} \tag{5}$$

$$P_{faco} = P_{faco}(0)P_{faco}(1) = (\frac{1}{2})(\frac{1}{2}) = \frac{1}{4} \tag{6}$$

Using conventional full-adders, larger counters have their switching activity at each intermediate output maximized at $\frac{1}{4}$, causing large power consumption. In our approach, we reduce the switching activity of individual modules, which has a significant effect on the overall average power consumption.

3. New Approach

Our proposed approach for (p, q) parallel counters uses 1-out-of-n code modules and encoders to produce the desired output. This approach was proposed in [5], and is shown in Figure 3.

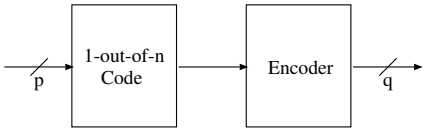

Figure 3. Stages of new approach

3.1. 1-out-of-n Code Stage

A general 1-out-of-n code module takes an initial p bits $\underline{x} = (x_{p-1}, x_{p-2}, \ldots, x_1, x_0)$ and produces a $p+1$ bit output $\underline{z} = (z_p, z_{p-1}, \ldots, z_1, z_0)$, where the value of z_i is given in Equation 7. This module is also referred to as a tally circuit [9].

$$z_i = \begin{cases} 1 \text{ if } \sum x_j = i \\ 0 \text{ otherwise} \end{cases} \tag{7}$$

A straightforward design for such a module consists of sums of minterms. For instance, considering $p = 3$, z_1 can be represented as: $z_1 = x_2' x_1' x_0 + x_2' x_1 x_0' + x_2 x_1' x_0'$, and designed with appropriate logic gates. However, this approach results in a topological kludge for large values of p. Considering $p = 7$, the representation of z_3 consists of ORing $\binom{7}{3} = 35$ minterms, each consisting of 7 literals.

A more structurally regular implementation for such a function, as described in [9] is an array of pass transistors. An example of a 2-to-3 tally circuit implemented in NMOS is shown in Figure 4. A $high$ signal propagates through the triangular array from the VDD source at the lower left side of the array. If x_i is high, the propagating signal travels to the next higher horizontal path. The total number of paths it travels is the number of inputs x_i that are high. Logic-0 signals propagate through the array from the ground points to all other outputs. The resulting output bits z_i form a 1-out-of-n code.

Our approach is based on this array tally circuit with the following modifications: (1) the n-type transistors are replaced with CMOS transmission gates; (2) the source and ground terminals are switched, with source directly connected to p-type transistors and ground directly connected to n-type transistors; (3) the output z_0, with its corresponding n-type transistor and transmission gate are removed. The purpose of (1) is to map the NMOS design of [9]

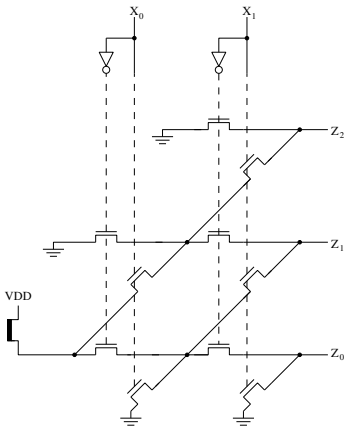

Figure 4. 2-to-3 tally module

into CMOS. The purpose of (2) is to generate inverted output signals $\overline{z_i}$ which will be passed to the encoding stage consisting of NAND-gates rather than OR-gates, due to the inherent faster speed, smaller cost, and lower power consumption of NAND-gates over OR-gates. The purpose of (3) is that output z_0 is not used at the encoding stage, and does not contribute to the one-set of the function.

A 2-to-2 tally circuit using these modifications is shown in Figure 5. As a note, inverting the outputs $\overline{z_2}$ and $\overline{z_1}$ produces an equivalent half-adder with sum bit z_1 and carry-out bit z_2. With each unit increase in p, the circuit extends to the right and upward one level. The 3-to-3 tally circuit illustrates this in Figure 6. The critical delay of a general p-to-p tally circuit can be characterized as the delay through p pass transistors, which is proportional to p^2 [9].

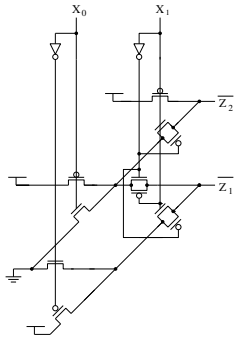

Figure 5. 2-to-2 Modified Tally Circuit

One drawback of simply adding another level though is that circuit does not provide a restoration of the logic levels, i.e. the logic gates are passive with no gain elements. A

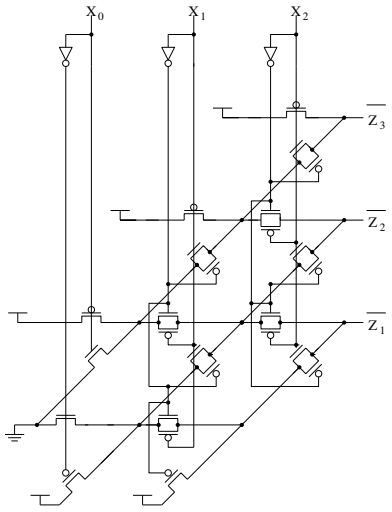

Figure 6. 3-to-3 Modified Tally Circuit

solution to this problem, as recommended in [1] is to insert inverters after every four transmission gates in series. The cost of a p-to-p tally circuit is a function of a $(p-1)$-to-$(p-1)$ tally circuit, with an additional level of logic. It can be characterized with the following recurrence, assuming p inputs, and inverters composed of two transistors:

$$C(p) = C(p-1) + 4(p+1) \tag{8}$$

which simplifies to the closed-form equation:

$$C(p) = 2p^2 + 6p - 5 \tag{9}$$

The switching activity of each output node $\overline{z_i}$ of a p-to-p tally circuit is the probability that the output switches from 0 to 1. The probability $P_{\overline{z_i}}(0)$ for such a circuit is:

$$P_{\overline{z_i}}(0) = \frac{\binom{p}{i}}{2^p} \tag{10}$$

$P_{\overline{z_i}}(0)$ is simply the probability that i among p inputs is 1, out of a total of 2^p possible input patterns. The switching activity is:

$$P_{\overline{z_i}} = P_{\overline{z_i}}(0)P_{\overline{z_i}}(1) = \left(\frac{\binom{p}{i}}{2^p}\right)\left(1 - \frac{\binom{p}{i}}{2^p}\right) \tag{11}$$

The values of $P_{\overline{z_i}}$ for various values of p are shown in Table 1.

3.2. Encoding Stage

An general (m, n) encoder takes an m-bit input $\underline{y} = (y_{m-1}, y_{m-2}, \ldots, y_1, y_0)$ and produces an n-bit output $\underline{z} =$

p	$P_{\overline{z_1}}$	$P_{\overline{z_2}}$	$P_{\overline{z_3}}$	$P_{\overline{z_4}}$	$P_{\overline{z_5}}$	$P_{\overline{z_6}}$	$P_{\overline{z_7}}$
2	0.250	0.187	—	—	—	—	—
3	0.234	0.234	0.109	—	—	—	—
4	0.187	0.234	0.187	0.058	—	—	—
5	0.131	0.214	0.214	0.131	0.030	—	—
6	0.084	0.179	0.214	0.179	0.084	0.015	—
7	0.051	0.137	0.198	0.198	0.137	0.051	0.007

Table 1. Switching Activity of Tally Modules

$(z_{n-1}, z_{n-2}, \ldots . z_1, z_0$, where $n = \lceil \log_2 m \rceil$, and $\underline{z}$ represents in binary code the index of the input i equal to 1. The conventional approach is to use appropriately sized OR-gates to produce the outputs. However, as mentioned before, we use inverted inputs and appropriately sized NAND-gates to produce the outputs, due to the benefits of NAND gates over OR gates, in terms of cost, delay, and power consumption. Also as noted, we only consider an $m-1$ input $y = (y_{m-1}, y_{m-2}, \ldots, y_1)$, eliminating y_0 due to its absence in producing the desired output.

The cost in terms of transistors, assuming m inputs is approximately $m \log_2 m$. The delay can be characterized as the delay of propagating through $\lceil m/2 \rceil$ transistors. The switching activity of encoders for various values of m is given in Table 2.

m	P_{z_0}	P_{z_1}	P_{z_2}
4	0.250	0.250	—
5	0.250	0.234	0.058
6	0.250	0.234	0.152
7	0.250	0.246	0.225
8	0.250	0.250	0.250

Table 2. Switching Activity of Encoders

4. Comparison of Approaches

We compare the new approach to the conventional approach for the implementation of various (p,q) counters, where p is in the range 2 to 7. We measure cost in terms of transistor count (TC), critical delay (Del) in nanoseconds, and average power consumption (Pwr) in milliwatts. Counters using the two methods were implemented conforming to MOSIS design rules, and were simulated using HSpice [10], with Vdd=3.3v, and temperature=25°C. An exhaustive test vector set was used for counters with $p \leq 4$, and 50 random test vectors were used for counters where $p > 4$. Transistion intervals were set at 10ns to accomodate the relatively large delays in producing the outputs of the (7,3) counters. The results are summarized in Table 3.

Counter	Conv			New		
	TC	Del	Pwr	TC	Del	Pwr
(2,2)	18	1.869	0.082	19	1.854	0.098
(3,2)	28	4.030	0.163	39	3.850	0.179
(4,3)	64	4.649	0.385	61	4.461	0.251
(5,3)	74	5.738	0.520	101	4.747	0.349
(6,3)	102	8.109	0.650	145	5.445	0.407
(7,3)	112	9.077	0.774	188	6.653	0.502

Table 3. Comparison of Approaches

5. Explanation of Results

The general increase in cost for the new approach over the conventional approach is due to the more exhaustive nature of the new approach. While the conventional approach reduces the number of bits from i to approximately $\lceil \frac{2}{3} i \rceil$ at each stage, the new approach produces p bits from the original p bits at the first stage, then reduces the p bits to $\lceil \log_2(p+1) \rceil$ at the encoding stage.

The overall reduction in delay is due to less transistors along the critical path for the new approach than the conventional approach, since the conventional approach uses inverters after each pass-transmission gate, whereas the new approach uses one buffer (two inverters) after every four transmission gates in series.

The overall reduction in average power consumption is due to less switching activity among the tally circuits and encoders within the new approach, than the full-adders and half-adders used in the conventional approach. This produces less glitching in the outputs, which helps significantly in reducing average power consumption.

6. Summary

In summary, we have presented a new approach for (p,q) counter implementations, based on using modules to produce 1-out-of-n codes, and encoders to produce the final output. The new approach reduces the critical delay and average power consumption, at a tradeoff of a generally larger cost in terms of transistors. This approach is scalable for larger counters, with each unit increase in p requiring one more level of logic, without greatly increasing the complexity of the circuit. This approach is attractive for use in implementing multipliers, multi-operand adders, and other arithmetic circuits, where reducing delay and power consumption is a major design aspect.

Acknowledgments. This research has been supported in part by the NSF Grant MIP-9314172 "Arithmetic Algorithms and Structures for Low-Power Systems."

References

[1] A. Bellaouar and M. Elmasry. *Low-Power Digital VLSI Design: Circuits and Systems*. Kluwer Academic Publishers, Norwell, Massachusetts, 1995.

[2] L. Bisdounis, G. Panagiotaras, O. Koufopavlou, and C. Goutis. CMOS multi-input gate implementations for low-power digital design. *International Journal of Electronics*, 79(5):641–653, November 1995.

[3] K. Current and D. Mow. Implementing parallel counters with four-valued threshold logic. *IEEE Transactions on Computers*, C-28(3):200–204, March 1979.

[4] L. Dadda. On parallel digital multipliers. *Alta Frequenza*, 45:574–580, October 1976.

[5] M. Ercegovac and T. Lang. A 16-to-5 counter for VLSI. *UCLA CSD, unpublished report*, September 1985.

[6] C. Foster and F. Stockton. Counting responders in an associative memory. *IEEE Transactions on Computers*, 20:1580–1583, 1971.

[7] H. Kobayashi and H. Ohara. A synthesizing method for large parallel counters with a network of smaller ones. *IEEE Transactions on Computers*, C-27(8):753–756, August 1978.

[8] LSI Logic Corporation. *LCA500K Preliminary Design Manual*. 1994.

[9] C. Mead and L. Conway. *Introduction to VLSI Systems*. Addison-Wesley, Philippines, 1980.

[10] Meta-Software. *HSpice User's Manual*. Campbell, CA, 1992.

[11] N. Nikityuk. Use of parallel counters for triggering. *Nuclear Instruments and Methods in Physics Research*, A321(3):571–582, Oct. 1992.

[12] E. Swartzlander. Parallel counters. *IEEE Transactions on Computers*, C-22:1021–1024, May 1973.

[13] D. Wong, G. DeMicheli, M. Flynn, and R. Huston. A bipolar population counter using wave pipelining to achieve 2.5x normal clock frequency. *IEEE Journal of Solid-State Circuits*, 27:745–753, May 1992.

[14] D. Zhang, G. Jullien, W. Miller, and E. Swartzlander. Arithmetic for digital neural networks. *10th Symposium on Computer Arithmetic*, pages 58–63, June 1991.