Designs of Counters with Near Minimal Counting/Sampling Period and Hardware Complexity

Chi-Hsiang Yeh
Dept. of Elec. & Computer Eng.
Queen's University
Kingston, ON K7L 3N6, Canada

Behrooz Parhami
Dept. of Elec. & Computer Eng.
University of California
Santa Barbara, CA 93106, USA

Yuke Wang
Dept. of Computer Sci.
University of Texas
Dallas, TX 75083, USA

Abstract

We propose several synchronous counter designs that have high counting and sampling rates and low cost at the same time. We first present carry-select counters which improve the maximum counting and sampling rates of previous counters based on carry anticipation by a factor of about 2, while requiring similar cost, or reduce the hardware cost of the fastest counters proposed thus far by a factor of about 2, while achieving comparable counting/sampling rate. We then propose a novel technique called postponed readout to further reduce the counting/sampling period to the delay of a 2-input AND gate plus the time for loading a flip-flop, while requiring similar cost. The resultant counting/sampling rate is competitive with the fastest previous designs and is achieved at a hardware cost that is lower by a factor of about 2. The price paid is that the count is read out 2 or 3 cycles later (depending on the length of the counter), instead of 1 cycle in previous synchronous counters.

1 Introduction

Counters are sequential circuits that keep track of the number of pulses applied on their inputs. They are among the most widely used components in digital systems, with applications in computer systems, communication equipments, scientific instruments, and industrial process control, to name a few. A vast variety of counter designs have been proposed in the literature [2, 8, 10, 11, 13, 14, 15, 17], patented [1, 3, 5], and/or used in practice [6, 9, 12]. They can be classified into synchronous counters, such as ring counters and twisted ring counters [6], and asynchronous counters, such as ripple counters [12]. In many applications, synchronous counters are required or preferred. Other criteria for the performance of counters include their counting rate, sampling rate and hardware cost.

Many conventional synchronous counter designs are based on an incrementer (i.e., an adder with addend 1) and a register [9, 12]. A vast variety of fast adders have been proposed in the literature [6, 7, 12, 16, 19, 21], which can be adapted for building fast incrementers and, thus, fast counters. However, full carry propagation is required in each increment cycle in such designs, leading to counting periods lower-bounded by $\Omega(\log m)$ for modulo-m counters, even when the fastest adders are used.

To further improve the performance, several constant-time counters have been proposed. Binary counters proposed by Vuillemin [18] have counting period equal to the delay of two 3-input gates plus the loading time of a flip-flop, and have cost equal to $\log_2 m$ flip-flops and half adders plus a small number of AND gates. They use about half as many flip-flops as the constant-time designs given in [4], but their maximum counting/sampling rates are only half those of the designs in [4], which have counting/sampling periods equal to the delay of an HA plus the loading time of a flip-flop.

In this paper, we first propose *carry-select counters*, whose counting/sampling rates are approximately double those of the designs proposed in [18], while achieving similarly low cost. We then propose a novel technique called *postponed readout* to further increase the counting and sampling rates. In a *synchronous counter with postponed readout*, the count read out is actually the total number of pulses up to several clock cycles ago, but it is synchronous in the sense that all count bits appear at the output at the same time so that we can continue inputting counting signals when reading out the count. We apply the technique to carry-select counters and obtain *carry-select counters with postponed readout (CSC/PRs)*, which further reduce the counting/sampling periods to the delay of a 2-input gate plus the loading time of a flip-flop. CSC/PRs are the fastest synchronous counters proposed in the literature thus far. The price paid is that the count of a CSC/PR appears at the output 2 or 3 cycles (depending on counter range) after the input is applied to the counter, instead of 1 cycle later. Carry-select counters and CSC/PRs are the only designs reported in the literature thus far that achieve near minimal count-

0-7803-6514-3/00/$10.00©2000 IEEE

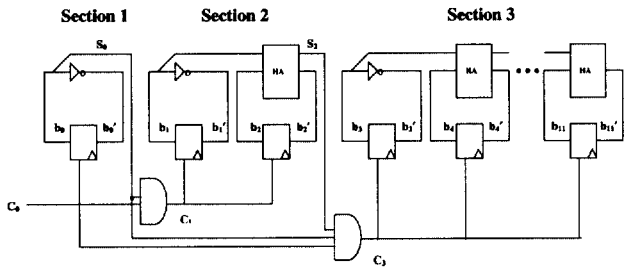

Figure 1. A 12-bit carry-select counter.

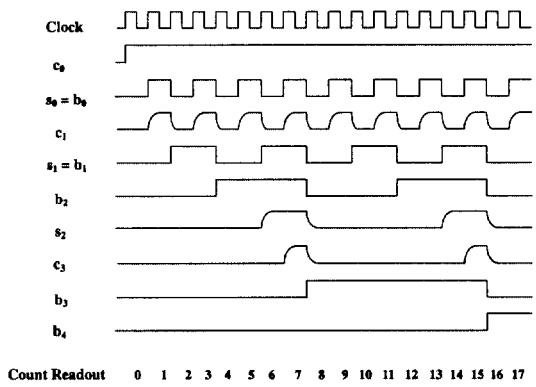

Figure 2. A timing diagram of the 12-bit carry-select counter. The control bits and the first 5 count bits of the counter are shown. The input signal is kept 1.

ing/sampling period and hardware cost at the same time. Our single-input counter designs can be combined with the techniques we proposed in [20] to derive efficient multi-input counters.

2 Carry-Select Counters (CSCs)

In this section we present carry-select counters, a variant of the binary counters proposed in [18], while increasing the maximum counting and sampling rate by a factor of about 2.

2.1 Short Carry-Select Counters

In Fig. 1 we present a 12-bit carry-select counter. The counter is composed of three sections with 1, 2, and 9 bits respectively, starting with the least significant bit for the count. If the number of bits required is smaller than 12, we simply remove flip-flops, and the associated HAs and AND gates, that are not needed. Each of the sections includes an incrementer, which is implemented as a carry-ripple adder with addend 1, the same as a carry-ripple counter with input 1 or a carry-anticipate counter [18]. The enable signal of each section is the logical AND of carry-out signals from all the previous sections.

The minimum counting period of carry-select counters is the delay of a 3-input AND gate plus the time required for loading a flip-flop. To verify the correctness of the counter in Fig. 1 for such a minimum counting period, we first consider the case where bit b_1 and possibly bit b_2 change at time 0. It can be seen that the earliest time for any bits in $b_3b_4b_5\cdots b_{11}$ to change again is time 2 since bits b_1 and b_2 will not change value before that, where a time unit is assumed to be equal to the counting period. Therefore, the requirement for carry c_3 is that it has to become stable before time 2. The signals corresponding to new values of c_0 and b_0 are fed directly to the AND gate for carry c_3 so they will not cause any problem. Let us now consider the signal corresponding to the new value of section-carry s_2. Section-carry s_2 will become stable before time 1, since the signals corresponding to the new values of b_1 and b_2 updated at time 0 only need to propagate through a 2-input AND gate; the signal corresponding to the new value of section-carry s_2 will in turn propagate through the 3-input AND gate and make carry c_3 stable before time 2. From this example, we can see that carry c_3 is always available in time after a change in b_1 and/or b_2.

Let us now examine whether b_{11} can be updated correctly. Consider another case where bit b_3 and an arbitrary subset of bits $b_4b_5b_6\cdots b_{11}$ change their values at time 100. It can be seen that the earliest time for any bits in $b_3b_4b_5\cdots b_{11}$ to change their values again is time 108 since bits b_3 will not change value before the counter receives another 8 inputs. Therefore, the requirement for b'_{11} is that it has to become stable before time 108. The signals corresponding to new values of $b_3b_4\cdots b_{11}$ will propagate through at most 7 concatenated 2-input AND gates and a 2-input XOR gate before time 108, so the new value of b'_{11} is ready to be loaded into the flip-flop for bit b_{11} (if required) at time 108. Note that if the signal can propagate through more than 7 concatenated 2-input AND gates and a 2-input XOR gate, we can add more bits to the third section of the counter without increasing the counting period. We illustrate the timing diagram of the 12-bit carry-select counter in Fig. 2. It can be seen that the carry-select counter functions correctly with the aforementioned counting period.

2.2 Long Carry-Select Counters

The design in Fig. 1 can be extended to obtain a k-bit carry-select counter for any $13 \le k \le 2^{31}+6$. Such a counter is composed of five sections with 1, 1, 3, 31, and $k-36$ bits

respectively for the count when $k \geq 39$, and is composed of four sections with 1, 1, 3, and $k-5$ bits respectively when $13 \leq k \leq 38$. Each of the sections includes an incrementer, and the enable signal of each section is the conjunction of the input c_0 and the section-carry bits from all the previous sections. The main difference between the counters in Fig. 1 and the extended design is that we now use more than one level of AND gates to keep their fan-ins equal to 3 or less. Signals c_0 and s_0 have to be fed directly to the AND gate at the last level since their values may change during every cycle. Other section-carry bits can be fed to the AND gate at the first level since the lengths of the corresponding sections are designed to be short enough for the section-carry bits to propagate through several levels of AND gates before the resultant carry bit is needed.

The minimum counting period of a long carry-select counter is still the delay of a 3-input AND gate plus the time required for loading a flip-flop. The correctness for the first three sections and bits $b_5 b_6 \cdots b_k$ can be verified as in the preceding examples. Let us now verify that carry bit c_5 will always become stable in time. Consider the case where bit b_2 and possibly bit b_3 and/or bit b_4 change at time 0. It can be seen that the earliest time for any bits in $b_5 b_6 \cdots b_{35}$ to change again is time 4 since bits $b_2 b_3 b_4$ will not change before that. Therefore, the requirement for carry bit c_5 is that it has to become stable before time 4. Section-carry s_4 will become stable before time 2, since the signals corresponding to the new values of $b_2 b_3 b_4$ updated at time 0 only need to propagate through at most two concatenated 2-input AND gates; the signal corresponding to the new value of section-carry s_4 will in turn propagate through two 3-input AND gates and make carry c_5 stable before time 4. The propagation of signals for all other inputs for computing carry bits can be verified in a similar manner.

A k-bit carry-select counter only requires k flip-flops, slightly fewer than k HAs, as well as several inverters and AND gates. The number of flip-flops required is approximately half that of the design proposed in [4]. Therefore, carry-select counters achieve near maximal count/sampling rate and near minimal hardware cost at the same time.

The design can be easily generalized to even longer counters and to carry-select counters with different section sizes which has important practical purposes. In general, as long as the length of a section is smaller than $2^{k'} - l + 1$ the counting rate remains the same, where k' is the total number of bits before the section and l is the maximum number of levels the section-carry of that section has to propagate to generate the carry bit(s) for subsequent section(s). For example, to obtain a 128 bit carry-select counter, we can use section sizes 1, 1, 3, 8, 115. An advantage for having more bits in the last section is that the subcircuit for the last section of such counters can use considerably slower logic (e.g., by a factor of 64) without reducing the counting rate. Such designs may lead to lower power and/or lower cost designs.

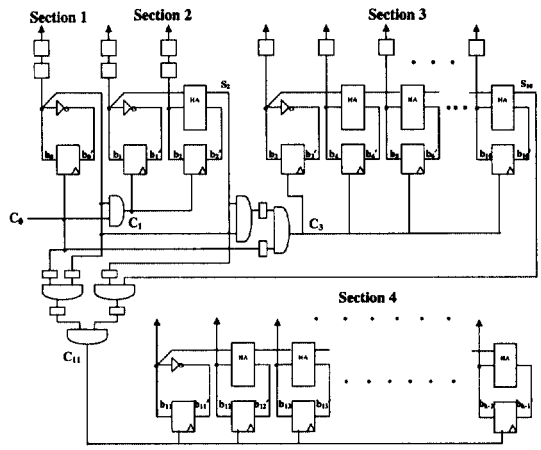

Figure 3. Design of a k-bit carry-select counter with postponed readout (CSC/PR) for any $4 \leq k \leq 2^{11} + 1$.

Note that in practice the length of a section can be larger than the preceding restriction since a 2-input AND gate requires delay smaller than a clock cycle. The large fan-out of the AND gate feeding larger sections can be reduced by replicating the AND gate. For example, the fan-out of the AND gate producing c_5 can be reduced from 31 to 8 by simply using four AND gate, each producing c_5 for 8 bits of the following section.

3 CSC with Postponed Readout

Carry-select counters with postponed readout (CSC/PRs) further reduce the counting period and sampling period to the delay of a 2-input gate plus the loading time of a flip-flop. However, the count read out from a CSC/PR is actually the total number of pulses up to several clock cycles ago. Therefore, carry-select counters with or without postponed readout are complementary and useful to different systems, depending on application requirements.

In Fig. 3 we present the design of k-bit CSC/PR for any $4 \leq k \leq 2^{11} + 1$. The counter is composed of four sections with 1, 2, 8, and $k-11$ bits for the count when $k \geq 13$, and is composed of three sections with 1, 2, and $k-3$ bits when $4 \leq k \leq 12$. Each of the sections also includes an incrementer, and the enable signal for each section is still the carry-out of all the previous sections. There are three major differences between the structures of carry-select counters and CSC/PRs: (1) Each bit of the first two sections of a CSC/PR will go through two levels of latches before read out, and each bit of the third section will go through one level of latches; (2) latches are added between two levels of AND gates for computing carry bits c_3 and c_{11}, and signals corresponding to c_0, s_0, and s_2 go through a level of latches before

being fed into the AND gates for computing carry c_{11}; (3) the count bits $b_0b_1b_2$ go through two levels of latches, and the count bits $b_3b_4\cdots b_{10}$ go through one level of latches, before read out.

The advantage of CSC/PRs is that its minimum counting period is reduced to the delay of a 2-input AND gate plus the time required for loading a flip-flop. The hardware cost of CSC/PRs is only slightly increased compared to the original carry-select counters since only a few additional latches are added to CSC/PR for the first few bits and the calculation of carry bits. Therefore, CSC/PR has the fastest counting and sampling rate among the synchronous counters proposed in the literature thus far, and requires near minimum hardware cost at the same time. Such counters, which we refer to as *synchronous counters with postponed readout*, provide the count as it was up to 3 cycles ago, and thus add up to 2 cycles to the usual 1 cycle delay.

Note that the sampling rates of carry-select counters, CSC/PRs, and synchronous counters with postponed readout in general are the same as their counting rate. Note also that the proposed postponed readout technique is different from the *delayed readout* mechanism we presented in [11] since the latter was mainly designed for combinational circuits and would lead to smaller sampling rate if applied to pipelined circuits.

The correctness for the count bits can be verified as the first example in Section 2. Let us now verify the correctness of carry bits in CSC/PRs. Consider the case where bit b_1 and possibly bit b_2 change at time 0. It can be seen that the earliest time for any bits in $b_3b_4b_5\cdots b_{10}$ to change again is time 2. Section-carry s_2 will become stable before time 1, and the corresponding signal will then propagate through the 2-input AND gate and be stored during time 2 to 3 at the latch before the input of the AND gate for carry bit c_3. This latched value together with the latched value for input c_0 from the previous cycle (time 1 to 2) will generate the value for carry bit c_3, which will be used to update the count bits $b_3b_4\cdots b_{10}$ at time 3. Note that this update is delayed by one clock cycle. Similarly, we can verify that carry bit c_{11} will become available to update the count bits with a delay of two cycles. Therefore, the count bits from the first two sections, which do not experience any additional delay, have to be delayed by two cycles before read out so that they are read at the same time as the corresponding count bits from the forth section. Similarly, the count bits from the third section have to be delayed by one cycle before read out. The sampling rate is not reduced due to the postponed readout since the count can be read during every clock cycle.

4 Multi-Input Counters

Fig. 4 illustrates the design for a k-bit multi-input counter [20]. The right part of the multi-input counter is a short accumulative parallel counter [11, 20], which is essentially a pipelined parallel counter followed by a pipelined ripple-carry adder. The full-adder sum and carry outputs are connected to latches; the carry out of the leftmost FA in each group that forms a ripple-carry adder is connected to a second latch before going to the next level. The left part of the multi-input counter is a carry-select counter (without postponed readout). We add 2 latches to the most significant sum bit of the pipelined ripple-carry adder in the last level, 3 latches to the second most significant sum bit, 4 latches to the third, and so on. Then the outputs of the carry-select counter and those of the last latches of the various sum bits collectively represent the count of the multi-input counter.

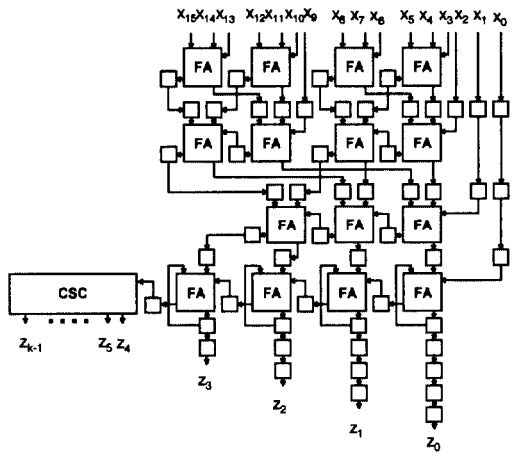

Figure 4. Design of a k-bit 16-input counter.

For an n-input counter, the count will be available for read out after $2\lceil\log_2 n\rceil$ cycles. For example, in the particular multi-input counter depicted in Fig. 4, the number of inputs is 16 so the count will appear at the lowermost sum latches and counter outputs after 8 cycles, no matter how large k is. Note that the readout delay $2\lceil\log_2 n\rceil$ is very close to the minimum possible. The counting and sampling rate of the multi-input counter in Fig. 4 is the delay of one FA (plus the loading time of a latch). Therefore, there is no need to use CSC/PR and a carry-select counter is usually fast enough for the left part of the design.

We may parallelize high-frequency sequential counting signals to allow the use of relatively slow, and thus low-cost, compact, and/or power-efficient, multi-input counters. For example, 800 MHz incoming signals, when demultiplexed 32 ways, are transformed into a set of 25 MHz signals that can be handled by non-speed-critical components. In this way, a complex circuit that would have to be designed by paying meticulous attention to speed optimization at every juncture (with the attendant overheads in design time, testing effort, VLSI area, and power consumption) is replaced by a simple high-speed front end feeding a lower-grade main part. With the unyielding pursuit of high-

throughput and low-power digital systems, this type of "demultiplexed" computation has become the norm in certain areas (notably data communications) that, as recently as a decade ago, used to rely on "multiplexed" hardware for reasons of economy.

By using a (twisted) ring counter, or another type of prescaler, as the first section and following it by a carry-select counter, we can obtain *ring-select counters* that have respective advantages. The postponed readout technique can also be incorporated into such designs. The details are omitted in this paper.

5 Conclusion

In this paper, we have presented counter designs based on carry-select and carry-select with postponed readout. These are the only counters reported in the literature thus far that achieve near maximal counting/sampling rate and near minimal hardware cost at the same time. Carry-select counters and CSC/PRs can be combined with short accumulative parallel counters to form long multi-input counters to implement counting with low power, low cost, or high performance. They can also be combined with ring counters or other prescalers. Moreover, the proposed postponed readout technique can be applied to many counter designs to obtain various synchronous counters with postponed readout.

References

[1] D. Chu and M. Ward, "Data capture in an uninterrupted counter," U.S. Patent No. 4,519,091, May 1985.

[2] D.W. Clark and L.-J. Weng, "Maximal and near-maximal shift register sequences: efficient event counters and easy discrete logarithms," *IEEE Trans. Computers*, Vol. 43, No. 5, pp. 560-568, May 1994.

[3] D.H. Eby, "Synchronous programmable two-stage serial/ parallel counter," U.S. Patent No. 4,905,262, Feb. 1990.

[4] M. Ercegovac and T. Lang, "Binary counter with counting period of one half adder independent of counter size," *IEEE Trans. Circuits and Systems*, Vol. 36, No. 6, pp. 924-926, Jun. 1989.

[5] M.W. Evans, "Minimal logic synchronous up/down counter implementations for CMOS," U.S. Patent No. 4,611,337, Sep. 1986.

[6] J.P. Hayes, *Introduction to Digital Logic Design*, Addison-Wesley, 1993.

[7] I. Koren, *Computer Arithmetic Algorithms*, Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[8] D.R. Lutz and D.N. Jayasimha, "Programmable modulo-K counters," *IEEE Trans. Circuits and Systems I: Fundamental Theory and Applications*, Vol. 43, No. 11, pp. 939-941, Nov. 1996.

[9] R.M.M. Oberman, *Counting and Counters*, Wiley, New York, 1981.

[10] B. Parhami, "Systolic up/down counters with zero and sign detection," *Proc. Symp. Computer Arithmetic*, pp. 174-178, 1987.

[11] B. Parhami and C.-H. Yeh, "Accumulative parallel counters," *Proc. Asilomar Conf. Signals, Systems, and Computers*, pp. 966-970, 1995.

[12] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*, Oxford University Press, 2000.

[13] K.Z. Pekmestzi and N. Thanasouras, "Systolic frequency dividers/counters," *IEEE Trans. Circuits and Systems II: Analog and Digital Signal Processing*, Vol. 41, No. 11, pp. 775-776, Nov. 1994.

[14] M.R. Stan, "Synchronous up/down counter with clock period independent of counter size," *Proc. IEEE Symp. Computer Arithmetic*, pp. 274-281, 1997.

[15] M.R. Stan, A.F. Tenca, and M.D. Ercegovac, "Long and fast up/down counters," *IEEE Transactions on Computers*, Vol. 47, No. 7, pp. 722-735, Jul. 1998.

[16] E.E. Swartzlander, Jr., *Computer Arithmetic*, Vol. I/II, IEEE Computer Society Press, Los Alamitos, CA, 1990.

[17] A.M. Tokarnia, "Identifying minimal shift counters: a search technique," *IEEE Trans. Computers*, Vol. 43, No. 5, pp. 633-639, May 1994.

[18] J.E. Vuillemin, "Constant time arbitrary length synchronous binary counters," *Proc. Int'l Symp. Computer Arithmetic*, pp. 180-183, 1991.

[19] C.-H. Yeh and B. Parhami, "Efficient pipelined multi-operand adders with high throughput and low latency: designs and applications," *Proc. Asilomar Conf. Signals, Systems, and Computers*, pp. 894-898, 1996.

[20] C.-H. Yeh and B. Parhami, "Efficient designs for multi-input counters," *Proc. 33th Asilomar Conf. Signals, Systems, and Computers*, Vol. 2, pp. 1340-1344, Oct. 1999.

[21] C.-H. Yeh, E. A. Varvarigos, B. Parhami, and H. Lee, "Optimal-depth circuits for prefix computation and addition," *Proc. 34th Asilomar Conf. Signals, Systems, and Computers*, 2000, to appear.