Comparison of Binary and LFSR Counters and Efficient LFSR Decoding Algorithm

Avinash Ajane, Paul M. Furth, Eric E. Johnson, and Rashmi Lakkur Subramanyam
Klipsch School of Electrical and Computer Engineering
New Mexico State University
Las Cruces, NM 88003, USA

Abstract— **This paper provides a direct comparison between a fast binary counter, built using a hierarchical Manchester carry chain, and a counter built using a linear feedback shift register (LFSR). The comparison is focused on speed, power and area consumption. We demonstrate the use of LFSRs as an alternative to conventional binary event counters. In order to use an LFSR as a counter, we developed an efficient algorithm for decoding the pseudo-random bit patterns of the LFSR counter to a known binary count. We implement 4-bit, 8-bit, 16-bit and 32-bit LFSR and binary counters in a 0.5-µm CMOS process. The hypotheses that LFSR counters leads to reduced area and higher speed were validated using simulation and measurement results.**

Index Terms— **Binary counters, LFSR counters, Manchester carry chain, Pseudo random number generator.**

I. Introduction

Due to the complexity involved in designing systems-on-a-chip (SoCs), manufacturing costs, testing time, and the amount of test data have all increased. It is found that linear feedback shift registers (LFSRs) help alleviate these three issues. For example, [1] introduces an SoC testing scheme in which the input test data are compressed using an LFSR. In [2], an LFSR is used for built-in self-test in the implementation of a Universal asynchronous receiver/transmitter on a field programmable gate array. It is also shown that binary counters used for an SoC consume more test time compared to the LFSR, when the input data set is large [2].

LFSRs are also widely used as event counters and efficient pseudo-random number generators [3]. For example, pseudo-random number generators can be used in cryptography to generate a secret key.

Binary counters generally use flip-flops, half adders, and a high-speed carry chain. The delay associated with a binary counter depends on the number of bits in the adder/carry chain circuit. In contrast, LFSR counters use only flip-flops and XOR gates. Their delay is independent of the number of bits in the counter.

In 1994, Intel's Pentium processor, implemented in a 0.6-µm process, ran at 100 MHz [4]. An on-chip event counter for a processor has a maximum allowable delay of one clock period, or 10 ns in this case. In a 0.6-µm process, the longest binary counter we expect to build with a maximum delay of 10ns is approximately 32 bits. In contrast, the maximum length of an LFSR counter we can build in the same process is much higher than 32 because the delay is independent of N.

The major drawback of using LFSR counters is the need to convert the pseudo-random LSFR count to a known binary count. The conversion can be performed using a hardware or software algorithm. Several algorithms exist to do this task. We introduce an algorithm that makes efficient use of memory and time in decoding LFSR counts to a known binary count.

In Section II we describe the design LFSR and binary counters. Simulation results of LFSR and binary counters are provided in the third section. In the fourth section, we discuss two algorithms that are frequently followed for conversion of a pseudo-random number to a known binary pattern and the new algorithm that we propose. We conclude with chip measurements, discussion, and conclusions.

II. Binary and LFSR Counter Design

The counters in this work are implemented using static logic gates. Dynamic logic, although occupying less area, requires periodic refreshing for long-term data retention [5]. However, for applications in the area of computer performance analysis (CPA), long time intervals between successive events could be encountered. One example from CPA is counting cache misses.

We added tri-state output buffers, select lines and a gated clock circuit to each counter, so that several counters could be fabricated on a single die. In general, all circuits were optimized for high speed operation.

A. Hierarchical Manchester Carry Chain Counter

As the name suggests, the high-speed counter in this work uses a hierarchical Manchester carry chain for carry propagation. This carry chain propagates the previous carry signal if the input signal A_i is high, and kills it otherwise. All the transmission gates (TGs) in Fig. 1 below have W_p/W_n = 6.0µm/3.0µm in order to reduce the propagation delay in the carry path. The pull-down nMOS transistors have W_n equal to

978-1-61284-857-0/11/$26.00

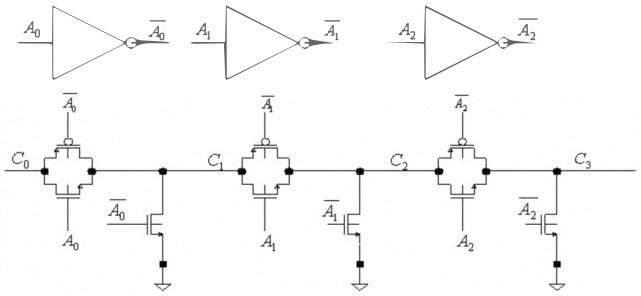

Figure 1. 4-bit static Manchester Carry Chain

1.5µm, while inverters have W_p/W_n equal to 3.0µm/1.5µm. Lengths of all devices are drawn as 0.6µm.

The schematic of the 4-bit counter is shown in Fig. 2. There are DFFs, XOR gates and a tri-state buffer (not shown) at each output. The counter has input carry C_0 at logic high, so that a count of one is added at every clock cycle.

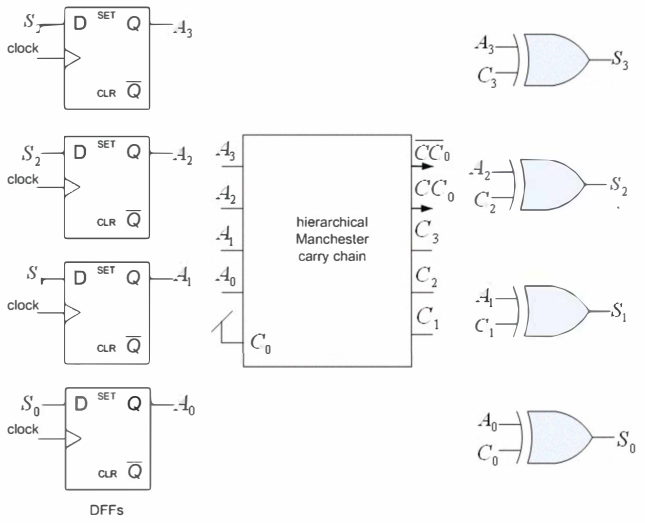

Figure 2. 4-bit binary counter schematic

The carry to the next hierarchal level will be high if and only if all the bits (A_0 : A_3) are high. A 4-input AND gate – with graded sizing of nMOS transistors in order to reduce the propagation delay [6] – is utilized to generate the carry skip signals CC_0.

This carry skip circuit is built into each 4-bit counter used in the longer counters.. The 8-bit counter consists of two 4-bit counters with a second level of hierarchy in the form of a 2-bit Manchester carry chain. The 16-bit counter is built with four 4-bit counters and a second level of hierarchy in the form of a 4-bit Manchester carry chain. The more complex 32-bit counter consists of four 8-bit counters linked by a third level of hierarchy in the form of a 4-bit Manchester carry chain. When necessary, inverting buffers were added to the carry chain so as to increase the operating frequency.

The complete binary counter chip consists of four counters of length 4, 8, 16, and 32 bits. A decoder is used to enable the outputs of only one counter at a time. A gated clock circuit enables only one counter at a time, so that other counters do not consume dynamic power. Fig. 3 shows the simulation results of this chip when the 4-bit counter is enabled.

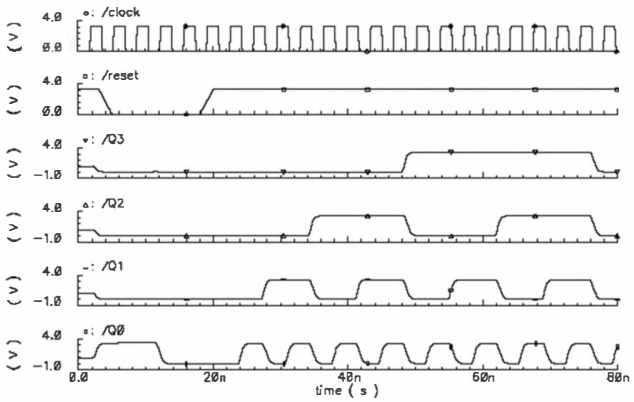

Figure 3. Transient response of the 4-bit counter at its maximum operating speed of 281MHz with V_{DD} = 3.3V

B. *The LFSR Counter*

The LFSR counter chip is similar to that of the binary counter chip, except that the reset signal is replaced by a set signal and there is no carry input signal. Whereas binary counters are reset to a value of 0, LFSR counters are set to a value of all 1's prior to event counting. If LFSR counters ever encounter the all 0's state, they will never get out of that state.

In this work, LFSR counters use XOR gates as feedback taps. When we detect high-frequency periodic patterns in the transient response, we know that the LFSR counter has failed.

The schematic of a 4-bit LFSR is shown in Fig. 4. The characteristic polynomial used for this LFSR is $1+x^3+x^4$, meaning that there is a feedback tap between the 3rd and 4th stages. Note that there is always feedback to the first stage. The characteristic polynomial used for the 8-, 16- and 32-bit LFSRs are $1+x^1+x^6+x^7+x^8$, $1+x^1+x^{12}+x^{15}+x^{16}$ and $1+x^1+x^{29}+x^{31}+x^{32}$, respectively [5], [7]. The transient simulation of the 4-bit LFSR counter is shown in Fig. 5.

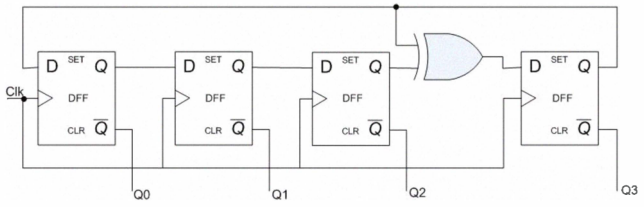

Figure 4. Schematic of the 4-bit LFSR

The polynomials given in [5], [7] are selected because they provide the maximum period before the pseudo-random pattern repeats, (2^N-1) cycles. The authors in [8] derive polynomials with a single tap for maximal or nearly maximal period, thereby increasing computational efficiency. In [9], a method to greatly enhance the maximal period of LFSRs using primitive polynomials is discussed.

III. COUNTER SIMULATION RESULTS

Simulations were carried out at V_{DD} = 3.3V and 1.1V for the two counters. The maximum operating frequency for each binary and LFSR-based counter is shown in Table I. From Table 1, we see that the speed-up factor of the LFSR counter over the binary counter is more than three for a 32-bit counter.

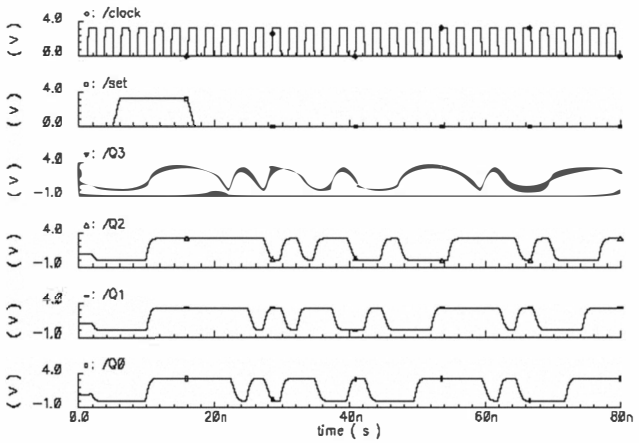

Figure 5. Simulated transient response of the 4-bit LFSR counter at its maximum operating speed of 403 MHz. V_{DD} = 3.3V

We also explored the average power consumption of the two counters, as shown in Table II. Counters were tested at the same operating speed and power supply voltage. From the table, we note that the power consumption of a 32-bit LFSR counter is 71% higher than that of a 32-bit binary counter. The reason for high power consumption in the LFSR is the large number of transitions.

TABLE I. SIMULATED MAXIMUM OPERATING FREQUENCIES OF BINARY AND LFSR EVENT COUNTERS WITH SPEED-UP FACTORS

Counter Length	f_{max} at V_{DD} = 3.3V [MHz]			f_{max} at V_{DD} = 1.1V [MHz]		
	Binary counter	*LFSR counter*	*Speed-up*	*Binary counter*	*LFSR counter*	*Speed-up*
4-bit	281	403	1.43 x	8.9	12.0	1.35 x
8-bit	209	375	1.79 x	6.5	11.5	1.77 x
16-bit	145	371	2.56 x	4.5	11.5	2.56 x
32-bit	113	373	3.30 x	3.7	11.6	3.14 x

TABLE II. SIMULATED AVERAGE POWER DISSIPATION OF BINARY AND LFSR EVENT COUNTERS WITH PERCENT CHANGE AT VDD = 3.3V

Counter Length	Operating Freq. [MHz]	Binary Counter [mW]	LFSR Counter [mW]	%change
4-bit	281	1.55	1.38	-11 %
8-bit	209	1.57	1.98	+26 %
16-bit	145	1.66	2.55	+54 %
32-bit	113	2.21	3.79	+71%

IV. LFSR DECODING ALGORITHM

To implement the LFSR as a counter, its pseudo-random pattern needs to be converted to a sequential binary count. There exist several algorithms to do this. One algorithm involves generating the LFSR output sequence, beginning with the all 1's state, and comparing the output with the input (the LFSR state for which the binary count is desired) at each loop iteration. The binary count in this case is given by the loop iteration number in which the match was found. The average number of comparisons, for an N-bit LFSR would then be $(2^N-1)/2 \approx 2^{N-1}$. For a 32-bit counter, the result is more than 2 billion comparisons, on average, each time an LFSR count must be converted to a binary count. This method is simple, and may be adequate for small counters. For example, decoding a 32-bit LFSR value requires about 12 seconds on a 900 MHz Itanium 2 processor. However, the decoding time increases rapidly with counter size: decoding a 40-bit LFSR count averages nearly 2 hours.

Another algorithm is one in which the loop iteration number of every LFSR state is stored in an array, indexed by that LFSR state. The input value for which we need to find the binary count is used as the index to a particular location in the array and the value stored at that location gives its binary count. The decoding time in this algorithm is negligible. However, the major drawback is that it uses a considerable amount of memory. For an N-bit LFSR, it occupies 2^{N-1} N-bit memory locations which amounts to 16 GBytes for a 32-bit LFSR. For a 40-bit LFSR, the memory requirement is 5 TBytes.

A. *Efficient Decoding Algorithm*

We propose an efficient decoding algorithm that effectively trades memory for speed. It was developed by one of the authors [10] and based on Hellman's Time-Memory Tradeoff algorithm for cryptanalysis [11]. For an N-bit LFSR, a hypothetical table of $2^{N/2}$ rows by $2^{N/2}$ columns is created. Each element of the table is labeled with a binary value sequentially from 0 to 2^N-2, leaving the first entry blank since the all 0's state is not permissible. The LFSR states are entered into the hypothetical table, in the order in which they appear. However, only the last column of this table is stored, along with the binary label, and the contents are sorted based on the LFSR states. Sorting is done in preparation for the fast binary search algorithm.

The LFSR input that needs to be decoded is searched in the stored column. If a match is found between the input and one of the states in the table, the corresponding binary label gives us the decoded binary count. If not, the LFSR input is cycled through the LFSR until it matches one of the entries in the column. The binary label of the matched state minus the number of cycles the LFSR needed to be run before a match is found gives the actual binary count. For this algorithm, an N-bit LFSR requires $2^{N/2+1}$ N-bit memory locations, and the average number of comparisons until a match is found is given by $2^{N/2-1} \log_2 2^{N/2}$.

B. *Decoding Algorithm Illustrated*

Table III helps to illustrate the proposed decoding algorithm applied to the 4-bit LFSR of Fig. 4. A hypothetical table of $2^{N/2}$ rows by $2^{N/2}$ columns is created, in this case only 4 by 4. The entry in each location includes the iteration number (0 to 14) and the corresponding LFSR state. Only the last column is stored. That column is then sorted based on the LFSR state, as shown in Table III.

Suppose we are asked to decode the LFSR state 0111. We use a fast binary search to look it up in the sorted column. However it is not found. We then cycle the LFSR to its next state, in this case, 1110. When we look up that value in the sorted column, we retrieve the iteration number of 2, and then subtract 1, the number of times we needed to cycle the LFSR. The decoded value is finally 1.

TABLE III. HYPOTHETICAL TABLE FOR 4-BIT LFSR WITH STORED AND SORTED COLUMNS

Iteration Number LFSR Binary Output				Stored Column	Sorted Column
-	0 1111	1 0111	2 1110	2 1110	10 0010
3 0101	4 1010	5 1101	6 0011	6 0011	6 0011
7 0110	8 1100	9 0001	10 0010	10 0010	14 1011
11 0100	12 1000	13 1001	14 1011	14 1011	2 1110

The advantage of this algorithm is that it is extremely fast compared to the first algorithm and consumes minimal memory as compared to second. In particular, for the 32-bit LFSR, it uses only 512 kBytes, versus 16 GBytes. Using the time-memory tradeoff approach, the average search time for a 32-bit LFSR value is 8 ms. And for a 40-bit value it is 1 s versus 2 hours. In general, this algorithm makes decoding LFSR counters sufficiently fast that their use becomes practical.

V. Counter Chip Measurement Results

The maximum operating frequencies of the counters at V_{DD} = 3.3V are higher than our currently available lab equipment. Therefore, we reduced the power supply voltage to 1.1V.

Table IV summarizes test results and layout areas for the conventional binary and LFSR counters. The 32-bit counter is not testable at V_{DD} = 1.1V because testing time would be prohibitively long.

A microphotograph of the LFSR counter chip is in Fig. 7.

TABLE IV. Measurements of Maximum Operating Frequency and Layout Area of Binary and LFSR Event Counters

Counter Length	f_{max} at V_{DD} = 1.1V [MHz]			Layout Area [mm^2]		
	Binary counter	*LFSR counter*	*Speed-up*	*Binary counter*	*LFSR counter*	*Area Saved*
4-bit	5.6	7	1.25 x	0.0100	0.0070	30 %
8-bit	4.1	6.7	1.63 x	0.0211	0.0146	31 %
16-bit	2.7	6.7	2.48 x	0.0431	0.0258	40 %
32-bit	-----	6.7	-----	0.0973	0.0510	48 %

VI. Discussion and Conclusions

Test results validate simulations, in that the LFSR counter was found to be faster than the conventional binary counter. Test results also show that as the length of the counter increases, the maximum operating frequency of the conventional counter decreases. On the other hand, increasing the length has negligible effect on the LFSR counter. Its maximum operating frequency remains almost constant at 6.7 MHz for a power supply of 1.1V, independent of *N*.

Comparing Tables I and IV, we see that the measured maximum operating frequencies are approximately 35% lower than simulation results. Several possible reasons for this disparity are supply noise, clock jitter, parasitic capacitances in the layout that are not accounted for during simulation, and normal process variations in threshold voltage and mobility.

This paper provided a comprehensive comparison of binary and LFSR counters in terms of speed, power, and area. The LFSR counter tends to have a much higher speed of

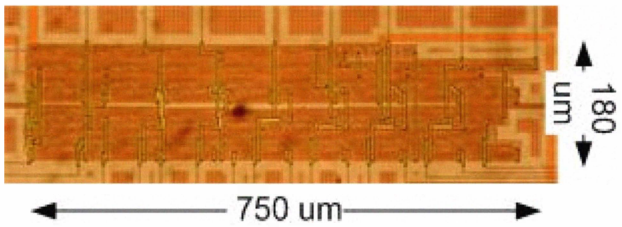

Figure 6. Microphotograph of LFSR counter chip consisting of 4-bit, 8-bit, 16-bit, and 32-bit LFSR event counters

operation. For a 16-bit counter, the measured maximum operating speed of the LFSR-counter was 2.5 times higher than the binary counter. Additionally, LFSR counters occupy significantly less area compared to the conventional binary counter, due to the simplicity in the LFSR logic. On the other hand, simulation results showed 54% higher power for a 16-bit LFSR counter and 71% higher power for a 32-bit LFSR counter, compared to their binary counterparts. The reason for high power consumption in the LFSR is the large number of transitions at each clock cycle.

Further, a conventional binary counter has the advantage of providing direct results, meaning no conversion of counts is needed. In this paper, we outlined an efficient algorithm for decoding the LFSR count to a known binary count.

References

[1] Z. Wang, K. Chakrabarty and S. Wang, "SoC testing using reseeding, and scan-slice-based TAM optimization and test scheduling," *Proceedings of Design Automation and Test in Europe*, 2007, pp. 201-206.

[2] S. Kakar, B. Singh and S. Khosla, "Implementation of BIST capability using LFSR tehniques in UART," *International Journal of Recent Trends in Engineering*, vol. 1, no. 3, May 2009, pp. 301-304.

[3] M. Dichtl, J. Dj. Golic, "High-speed true random number generation with logic gates only," *in Cryptographic Hardware and Embedded Systems*, Springer, September 2007, vol. 4727, pp. 45-62.

[4] Microprocessor Quick Reference Guide (2008). Retrieved January 2010, from Intel.com Website: http://www.intel.com/pressroom/kits/quickreffam.htm#pentium

[5] N. H. E. Weste and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*, 3rd edition, Pearson Education, Boston, 2005.

[6] S. Kang and Y. Leblebici, *CMOS Digital Integrated Circuits: Analysis and Design*, 3rd edition, McGraw-Hill, New York, 2003.

[7] Linear Feedback Shift Registers. Retrieved January 2010, from New Wave Instruments Website: http://www.newwaveinstruments.com/resources/articles/m_sequence_linear_feedback_shift_register_lfsr.htm.

[8] D. W. Clark and L. Weng, "Maximal and near- maximal shift register sequences: efficient event counters and easy discrete logarithms," *IEEE Transactions on Computers*, vol. 23, May 1994, pp. 506-568.

[9] A. Molina-Rueda, F. Uceda-Ponga and C. Feregrino Uribe, "Extended period LFSR using variable TAP function," *18th International Conference on Electronics, Communication and Computers*, 2008, pp. 129-132.

[10] Personal communication, E.E. Johnson, February 16, 2005 Department of Electrical & Computer Engineering, New Mexico State Univ.

[11] M. Hellman, "A cryptanalytic time-memory tradeoff," *IEEE Transactions on Information T heory*, vol. 26, 1980, pp. 401-406.