Efficient Shift Registers, LFSR Counters, and Long Pseudo-Random Sequence Generators

August 1995

Application Note By PETER ALFKE

Summary

Shift registers longer than eight bits can be implemented most efficiently in XC4000E RAM. Using Linear Feedback Shift-Register (LFSR) counters to address the RAM makes the design even simpler. This application note describes 4- and 5-bit universal LFSR counters, very efficient RAM-based 32-bit and 100-bit shift registers, and pseudo-random sequence generators with repetition rates of thousands and even trillions of years, useful for testing and encryption purposes. The appropriate taps for maximum-length LFSR counters of up to 168 bits are listed.

Xilinx Family

XC4000E

Demonstrates

Shift registers implemented in RAM, LFSR counters.

Introduction

The XC4000E on-chip distributed synchronous RAM architecture lends itself well to the efficient implementation of long shift registers. The 16 x 1 or 32 x 1 RAM behaves like an edge-triggered register. An address counter supplies sequential addresses, but there is no need for a conventional binary address sequence. Any repetitive pattern is acceptable, and a linear feedback shift register counter is the most efficient. In the examples below the conventional LFSR counter algorithm has been modified to guarantee no lock-up, even in the all-ones state.

Note that the established literature describes the outputs of LFSRs as Q1 to Qn (not Q0 to Qn-1, as is customary in binary counters). In order to be consistent with prior literature, LFSR bits are therefore labeled 1 to n thoughout this application note.

For a 4-bit counter, the basic XNOR feedback from Q3 and Q4 would exclude the all-ones state. By decoding the two states where the lower three bits are all ones, and inverting the feedback for those states, the 4-bit LFSR counter counts module 16, and has no lock-up state. Counters with a shorter cycle require additional decoding of the feedback signal, as shown in Table 1 and Figure 1. Any such decoding is easily done in the front-end CLB function generator. For a 5-bit counter, the following page shows the connections required for dividing by any number up to 32.

Divide-By 5 to 16 Counter in Two CLBs

Feedback for ÷16: (Q3 XNOR Q4) XOR
(Q1 AND Q2 AND Q3)

To divide by a number smaller than 15, use AND gate “C” to decode the binary pattern listed next to the desired number. For ÷15 and any number listed to the right of the & symbol, also add Q4 to AND gate “B”, thus skipping the all-ones state. All these counters avoid lock-up in the all-ones state.

1234	
0000	
1000	7
1100	11
1110	
1111	
0111	13 & 12
1011	10
1101	5 & 4

1234	
0110	3
0011	11
1001	9 & 8
0100	14
1010	6
0101	2
0010	5
0001	

for ÷16: do not connect Q4 to AND gate “B”, do not use AND gate “C”
for ÷15: connect Q4 to AND gate “B”, do not use AND gate “C”
for ÷<15: program AND gate “C" according to the table
for ÷ 4, 8,12,15: connect Q4 to AND gate “B”
for all other numbers: do not connect Q4 to AND gate “B”.

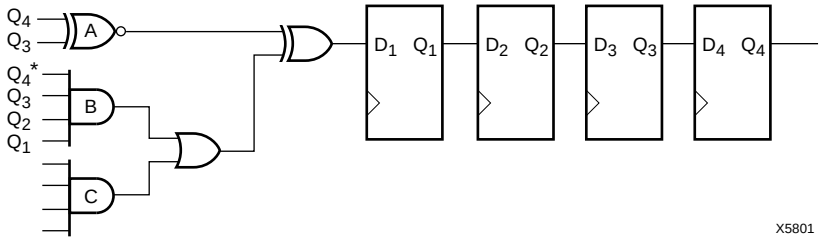

Figure 1. Divide by 5 to 16 Counter

Divide-By 2 to 32 Counter in 2.5 CLBs

Feedback for ÷32: (Q3 XNOR Q5) XOR
(Q1 AND Q2 AND Q3 AND Q4)

To divide by a number smaller than 31, use AND gate “C” to decode the binary pattern listed next to the desired number. For ÷31 and any number listed to the right of the & symbol, also add Q5 to AND gate “B”, thus skipping the all-ones state. All these counters avoid lock-up in the all-ones state.

for ÷32: do not connect Q5 to AND gate “B”, do not use AND gate “C”

for ÷31: connect Q5 to AND gate “B”, do not use AND gate “C”

for ÷<31: program AND gate “C”according to the table

for ÷5,10,14,16,18, 22, 27, 31: connect Q5 to AND gate “B”

for all other numbers: do not connect Q5 to AND gate “B”.

12345	
00000	
10000	23 & 22
11000	7
11100	19 & 18
01110	17
00111	8
10011	12
01001	28 & 27
00100	15
00010	11
10001	9
01000	4
10100	30
01010	21
10101	2
11010	26
11101	13
11110	31
11111	
01111	15 & 14
10111	29
11011	6 & 5
01101	26
10110	3
01011	11 & 10
00101	17 & 16
10010	20
11001	25
01100	6
00110	24
00011	21
00001	31

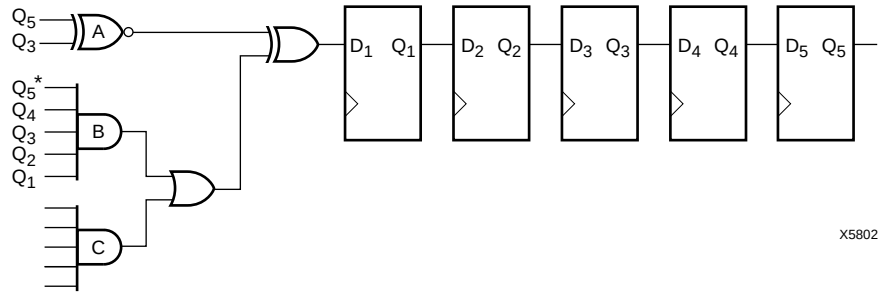

Figure 2. Divide by 2 to 32 Counter

RAM-Based Shift Registers

As shown in Flgure 3, a 32 x 1 shift register design requires two CLBs for the ÷16 address counter plus one CLB for the RAM. An 8-bit wide, 32-bit long shift register would use seven additional CLBs for RAM storage and output registers. Wider and longer shift registers can easily be implemented using the same concept. For increased length, it is most efficient to divide the length into equal parts of up-to 16 bits each and use a common address counter.

Figure 4 shows a 100-bit long, 8-bit wide shift register as an example. It uses two CLBs to implement a divide-by 16 counter, plus 24 CLBs for RAM storage and additional registers. Each bitstream uses three cascaded CLBs with their RAMs acting as 16+16 bit registers, plus four of their flip-flops used to bring the total shift-register length to 100. This design thus emulates 800 bits of shift register in only 26 CLBs, and it can run at a 70 MHz clock rate. Traditional register-based designs would use 5,600 equivalent gates for this complete function (seven gates per register bit). Here it occupies 6.5% of an XC4010E. Does that qualify the XC4010E as an 86,000 gate device ?

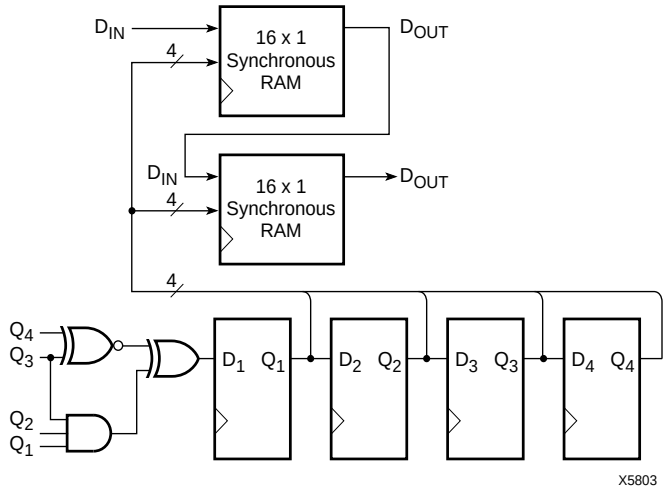

Figure 3. 32 x 1 Shift Register in 3 CLBs

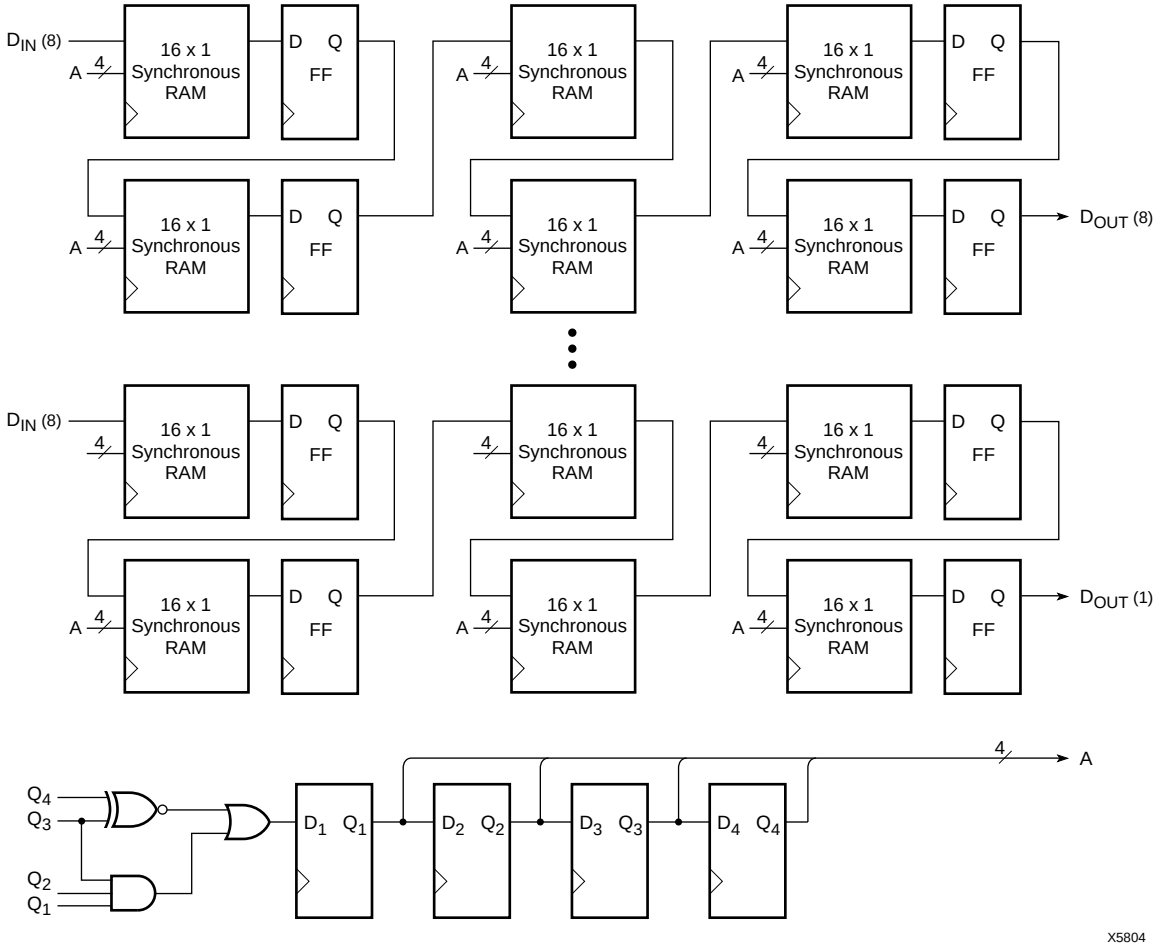

Figure 4. 100 x 8 Shift Register in 26 CLBs

Pseudo-Random Sequence Generator in Four CLBs

Any long LFSR counter generates a long pseudo-random sequence of zeros and ones. The sequence is not exactly random since it repeats eventually, and it also follows a mathematically predictable sequence. But for most practical purposes it can be considered random.

A 63-bit LFSR counter has a repetition time of (2^{63}-1) clock periods. Running at 50 MHz, such a counter repeats after more than five thousand years (5,849 years to be more precise), which is long enough to be irrelevant for most practical purposes. Conceptually, a 63-bit LFSR counter consists of a 63-bit shift register, with an XNOR feedback from the last stage and the next-to-last stage. The 63-bit length was actually chosen because of its conveniently simple feedback. Other maximum-length LFSR counters require different XNOR feedback taps. The appendix describes the maximum-length feedback connections for all LFSR counters of up to 168 bits in length.

The conventional shift register implementation of a 63-bit LFSR counter requires 32 CLBs in XC3000 or XC4000 family devices. By using a RAM-based approach, only two CLBs are needed, plus the addressing counter, which can be a ÷15 LFSR counter in two CLBs. With proper partitioning, the complete 63-bit pseudo-random sequence generator shown in figure 5 requires only four CLBs, and is capable of running at up to 70 MHz. A starting pattern of up to 63 bits can be flrst loaded into the shift register, and the output then generates a pseudo-random sequence of zeros and ones. The design can be expanded to a 127-bit LFSR counter in six CLBs, or a 159-bit LFSR counter in seven CLBs. Either of these two counters has a repetition period many billion times longer than the life of the universe.

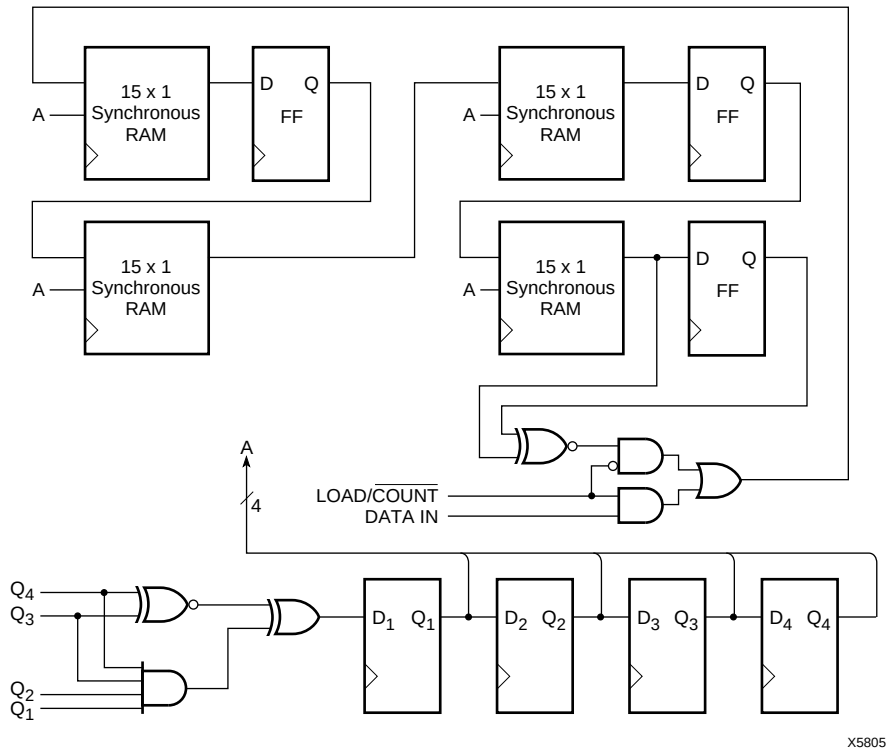

Figure 5. 63-Bit LFSR Counter in 4 CLBs

Linear Feedback Shift Register Taps

This table lists the appropriate taps for maximum-length LFSR counters of up to 168 bits. The basic description and the table for the first 40 bits was originally published in XCELL and reprinted on page 9-24 of the 1993 and 1994 Xilinx Data Books.

Responding to repeated requests, the list is here extended to 168 bits. This information is based on unpublished research done by Wayne Stahnke while he was at Fairchild Semiconductor in 1970.

n	XNOR from
3	3,2
4	4,3
5	5,3
6	6,5
7	7,6
8	8,6,5,4
9	9,5
10	10,7
11	11,9
12	12,6,4,1
13	13,4,3,1
14	14,5,3,1
15	15,14
16	16,15,13,4
17	17,14
18	18,11
19	19,6,2,1
20	20,17
21	21,19
22	22,21
23	23,18
24	24,23,22,17
25	25,22
26	26,6,2,1
27	27,5,2,1
28	28,25
29	29,27
30	30,6,4,1
31	31,28
32	32,22,2,1
33	33,20
34	34,27,2,1
35	35,33
36	36,25
37	37,5,4,3,2,1
38	38,6,5,1
39	39,35
40	40,38,21,19
41	41,38
42	42,41,20,19
43	43,42,38,37
44	44,43,18,17
45	45,44,42,41
46	46,45,26,25
47	47,42
48	48,47,21,20
49	49,40
50	50,49,24,23
51	51,50,36,35
52	52,49
53	53,52,38,37
54	54,53,18,17
55	55,31
56	56,55,35,34
57	57,50
58	58,39
59	59,58,38,37
60	60,59
61	61,60,46,45
62	62,61,6,5
63	63,62
64	64,63,61,60
65	65,47
66	66,65,57,56
67	67,66,58,57
68	68,59
69	69,67,42,40
70	70,69,55,54
71	71,65
72	72,66,25,19
73	73,48
74	74,73,59,58
75	75,74,65,64
76	76,75,41,40
77	77,76,47,46
78	78,77,59,58
79	79,70
80	80,79,43,42
81	81,77
82	82,79,47,44
83	83,82,38,37
84	84,71
85	85,84,58,57
86	86,85,74,73
87	87,74
88	88,87,17,16
89	89,51
90	90,89,72,71
91	91,90,8,7
92	92,91,80,79
93	93,91
94	94,73
95	95,84
96	96,94,49,47
97	97,91
98	98,87
99	99,97,54,52
100	100,63
101	101,100,95,94
102	102,101,36,35
103	103,94
104	104,103,94,93
105	105,89
106	106,91
107	107,105,44,42
108	108,77
109	109,108,103,102
110	110,109,98,97
111	111,101
112	112,110,69,67
113	113,104
114	114,113,33,32
115	115,114,101,100
116	116,115,46,45
117	117,115,99,97
118	118,85
119	119,111
120	120,113,9,2
121	121,103
122	122,121,63,62
123	123,121
124	124,87
125	125,124,18,17
126	126,125,90,89
127	127,126
128	128,126,101,99
129	129,124
130	130,127
131	131,130,84,83
132	132,103
133	133,132,82,81
134	134,77
135	135,124
136	136,135,11,10
137	137,116
138	138,137,131,130
139	139,136,134,131
140	140,111
141	141,140,110,109
142	142,121
143	143,142,123,122
144	144,143,75,74
145	145,93
146	146,145,87,86
147	147,146,110,109
148	148,121
149	149,148,40,39
150	150,97
151	151,148
152	152,151,87,86
153	153,152
154	154,152,27,25
155	155,154,124,123
156	156,155,41,40
157	157,156,131,130
158	158,157,132,131
159	159,128
160	160,159,142,141
161	161,143
162	162,161,75,74
163	163,162,104,103
164	164,163,151,150
165	165,164,135,134
166	166,165,128,127
167	167,161
168	168,166,153,151

LFSR Counters, 3 to 168 Bits

Conventional binary counters use complex or wide fan-in logic to generate high end carry signals. A much simpler structure sacriflces the binary count sequence, but achieves very high speed with very simple logic, easily packing two bits into every CLB. Such Linear Feedback Shift-Register (LFSR) counters are also known as pseudo-random sequence generators.

An n-bit LFSR counter can have a maximum sequence length of 2^n-1. In that case, it goes through all possible code permutations except one, which would be a lock-up state. A maximum length n-bit LFSR counter consists of an n-bit shift register with an XNOR in the feedback path from the last output Qn to the flrst input D1. The XNOR makes the lock-up state the all-ones state; an XOR would make it the all-zeros state. For normal Xilinx applications, all-ones is more easily avoided, since “by default” the flip-flops wake up in the all-zeros state. The table on page 5 describes the outputs that must be used as inputs of the XNOR. LFSR outputs are traditionally labeled 1 through n, with 1 being the first stage of the shift register, and n being the last stage. This is different from the conventional 0 to (n-1) notation for binary counters. A multi-input XNOR is also known as an even-parity circuit. Note that the connections described in this table are not necessarily unique; certain other connections may also result in maximum length sequences.

Examples

- A 10-bit shift register counts modulo 1023, if the input D1 is driven by the XNOR of Q10 and the bit three positions to the left (Q7), i.e. a one is shifted into D1 when Q10 and Q7 have even parity, which means they are identical.
- An 8-bit shift register counts modulo 255 if the input D1 is driven by the XNOR of Q8, Q6, Q5, Q4, i.e., a one is shifted into D1 if these four outputs have even parity, (four zeros, or two ones, or four ones).

References:

Wayne Stahnke, *Primitive Polynomials Modulo Two*, private communication in 1970, giving the following references:

E.R. Berlekamp, *Algebraic CodingTheory*, McGraw-Hill, 1968

P.H.R. Scholefield, *Shift Registers Generating Maximum-Length Sequences,* Electronic Technology, 10-1960, pp 389-394

S.W. Golomb, *Shffl Register Sequences*, Holden-Day, San Francisco, 1967

E.J. Watson, *Primitive Polynomials (Mod 2)*, Math. Comp. v.16 pp. 368, 1962

N. Zierler and J. Brillhart, *On Primitive Trinomals*, Information and Control v. 13, pp 541-554, 1968, and v. 14, pp 566-569, 1969

R.W. Marsh, *Table of Irreducible Polynomials*, Dept. of Commerce, October 1957

H. Riesel, *En Bok om Primtal,* Studentlitteratur, Denmark, 1968

M. Kraitchik, *Théorie dés Nombres*, Gauthier-Villars, Paris, 1922 and 1952

M. Kraitchik, *On the Factorization of* $2^n \pm 1$, Scripta Mathematica, v. 18, pp. 39-52, 1952

J. Brillhart, *Miscellaneous Factorizations*, Math. Comp., v. 17 pp. 447-450, 1963

J. Brillhart and J.L.Selfridge, *Factorizations...*, Math. Comp., v. 21, pp. 87-96, 1967

K.R. Isemonger, *Additional Factorizations...*, Math. Comp., v. 19, pp. 145-146, 1965

R.M. Robinson, *Some Factorizations...*, Math. Comp., v. 11, pp. 265-268, 1957

XILINX®
The Programmable Logic Company℠

Sales Offices

Corporate Headquarters
Xilinx, Inc.
2100 Logic Drive
San Jose, CA 95124 U.S.A.
Tel: 1 (408) 559-7778
FAX: 1 (408) 559-7114

Europe
Xilinx, Ltd.
Suite 1B, Cobb House
Oyster Lane
Byfleet
Surrey KT14 7DU
United Kingdom
Tel: 44-1932-349401
FAX: 44-1932-349499

Japan
Xilinx K. K.
Daini-Nagaoka Bldg. 2F
2-8-5, Hatchobori Chuo-ku.
Tokyo 104, Japan
Tel: (81) 3-3297-9191
FAX: (81) 3-3297-9189

Printed in U.S.A.

PN 001xxx-xx