What's an LFSR?

SCTA036A
December 1996

TEXAS INSTRUMENTS

Contents

List of Illustrations

Introduction

The purpose of this article is to explain what a Linear Feedback Shift Register (LFSR) and a Parallel Signature Analyzer (PSA) are and how to use them to test a TI Application-Specific Integrated Circuit (ASIC) using SCOPE™ cells. This article begins with a description of an LFSR, goes into Pseudorandom Pattern Generation (PRPG) and fault grading, describes a PSA, and, last, shows how to implement the PSA and LFSR functions using SCOPE boundary-scan cells, which are compatible with IEEE 1149.1.

LFSR

An LFSR is a shift register that, when clocked, advances the signal through the register from one bit to the next most-significant bit (see Figure 1). Some of the outputs are combined in exclusive-OR configuration to form a feedback mechanism. A linear feedback shift register can be formed by performing exclusive-OR on the outputs of two or more of the flip-flops together and feeding those outputs back into the input of one of the flip-flops as shown in Figure 2.

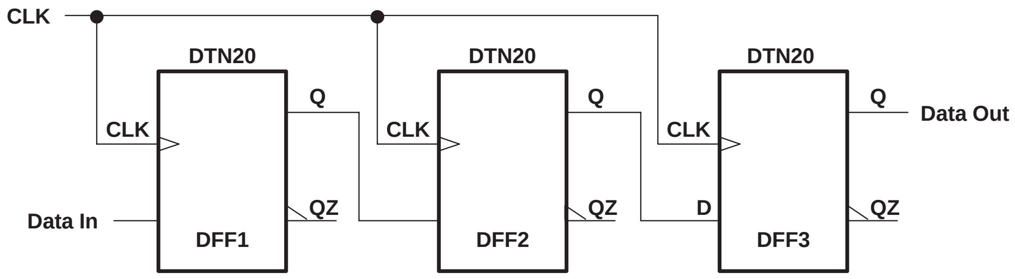

Figure 1. A 3-Bit Shift Register

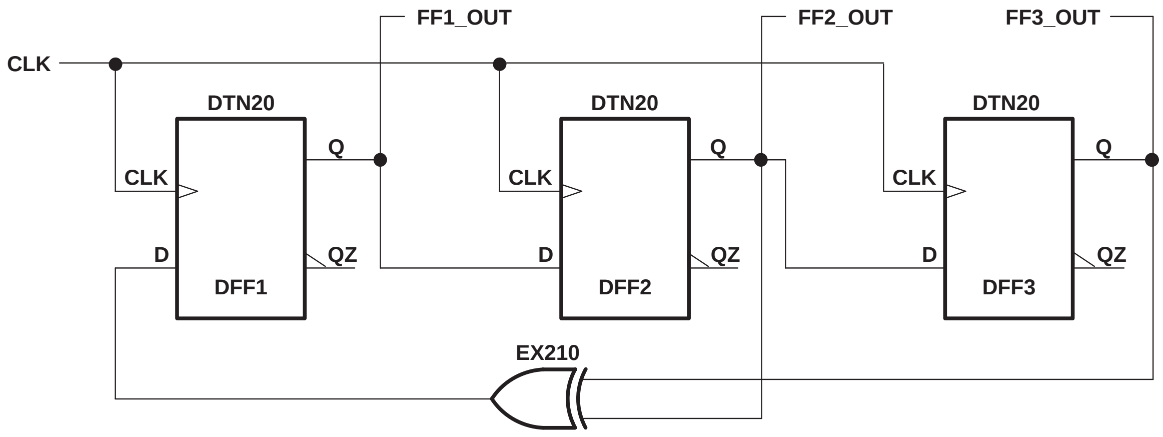

Figure 2. Linear Feedback Shift Register

Pseudorandom Pattern Generation

Linear feedback shift registers make extremely good pseudorandom pattern generators. When the outputs of the flip-flops are loaded with a seed value (anything except all 0s, which would cause the LFSR to produce all 0 patterns) and when the LFSR is clocked, it will generate a pseudorandom pattern of 1s and 0s. Note that the only signal necessary to generate the test patterns is the clock.

SCOPE is a trademark of Texas Instruments Incorporated.

Maximal-Length LFSRs

A maximal-length LFSR produces the maximum number of PRPG patterns possible and has a pattern count equal to $2^n - 1$, where n is the number of register elements in the LFSR. It produces patterns that have an approximately equal number of 1s and 0s and have an equal number of runs of 1s and 0s.[1]

Because there is no way to predict mathematically if an LFSR will be maximal length, Peterson and Weldon[2] have compiled tables of maximal-length LFSRs to which designers may refer. Table 1 shows the patterns produced by the LFSR in Figure 2, assuming that a pattern of 111 was used as a seed.

Table 1. Pattern-Generator Seed Values

CLOCK PULSE	FF1_OUT	FF2_OUT	FF3_OUT	COMMENTS
1	1	1	1	Seed value
2	0	1	1	
3	0	0	1	
4	1	0	0	
5	0	1	0	
6	1	0	1	
7	1	1	0	
8	1	1	1	Starts repeat

In a practical ASIC design, the user would create an LFSR that is much bigger than three bits to get a large number of pseudorandom patterns before the patterns repeated. However, there are some practical restrictions to the length of the LFSR. A 32-bit maximal-length LFSR would create over 4 billion patterns that, at a 16-MHz clock rate, would take almost 5 minutes to generate the whole pattern set.

PRPG and Fault Grading

The LFSR and PRPG techniques are often used to create functional patterns that provide a high level of fault coverage for the ASIC with minimum effort by the designer or the test engineer. Pseudorandomly generated patterns have been proven to very quickly generate high-fault-coverage results. The PRPG technique works especially well for combinational logic but may also work well for certain cases of sequential circuits because each input signal stimulated by the LFSR is frequently changing from a 1 to 0 and back again (high bit-toggle rate).

A designer would design the ASIC circuit, then simulate the design to verify correct functionality and timing. Once verified, the LFSR is designed and the outputs of the LFSR are connected to the ASIC's inputs – one LFSR output for each ASIC input. Figure 3 shows how the LFSR outputs are multiplexed with the ASIC inputs so that the ASIC application logic can be stimulated by either the normal data inputs or by the LFSR outputs. Note that no extra pins are required to implement the LFSR.

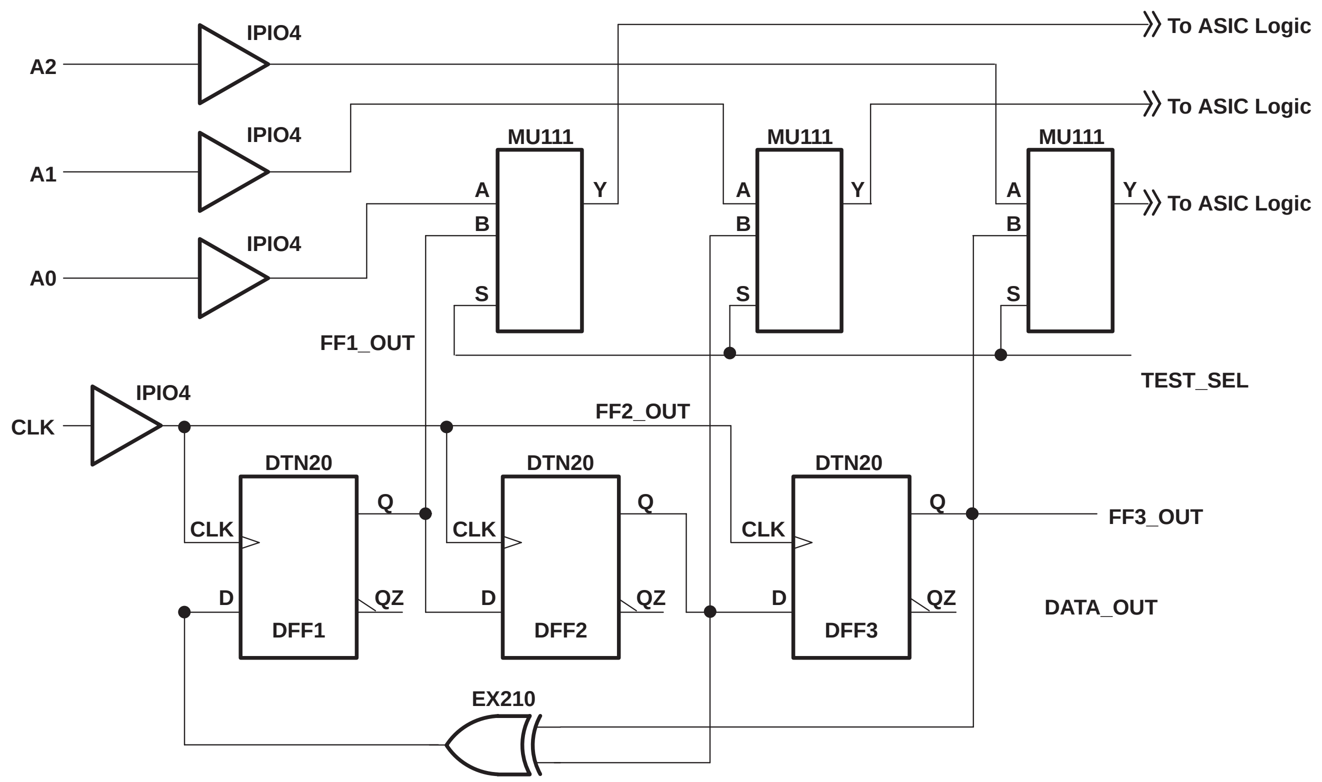

Figure 3. LFSR With Outputs Multiplexed With ASIC Inputs

Typically, a designer would capture the LFSR (or PSA) and simulate it by itself to evaluate and verify this subcircuit without having to simulate the total design. The designer could then easily examine the effects of different seed values on the patterns produced. This is particularly important if not all the $2^n - 1$ patterns that the LFSR could produce will be used in circuit testing. A designer would typically print out a sample of the patterns (every tenth or hundredth pattern, etc.) and use the printout when simulating the LFSR in the complete circuit to verify that the LFSR is generating the correct signals. Since the logic of the circuit has already been verified, the device would be put in the test mode to let the LFSR generate the inputs to the ASIC. Next, the outputs would be sampled every clock cycle to generate the expected outputs, given the inputs that the LFSR has created.

After verifying that the LFSR is generating the correct circuit inputs, the designer uses the resulting simulation vectors to fault grade the design. Fault grading enables the customer to ensure that the test vector set supplied for the design will catch manufacturing defects, such as shorted transistors and open metal lines (stuck-at faults, SA0, SA1). If the fault grade that results from applying the LFSR-generated simulation pattern is lower than required, the designer must generate additional patterns to raise the fault-grade level to the required value.

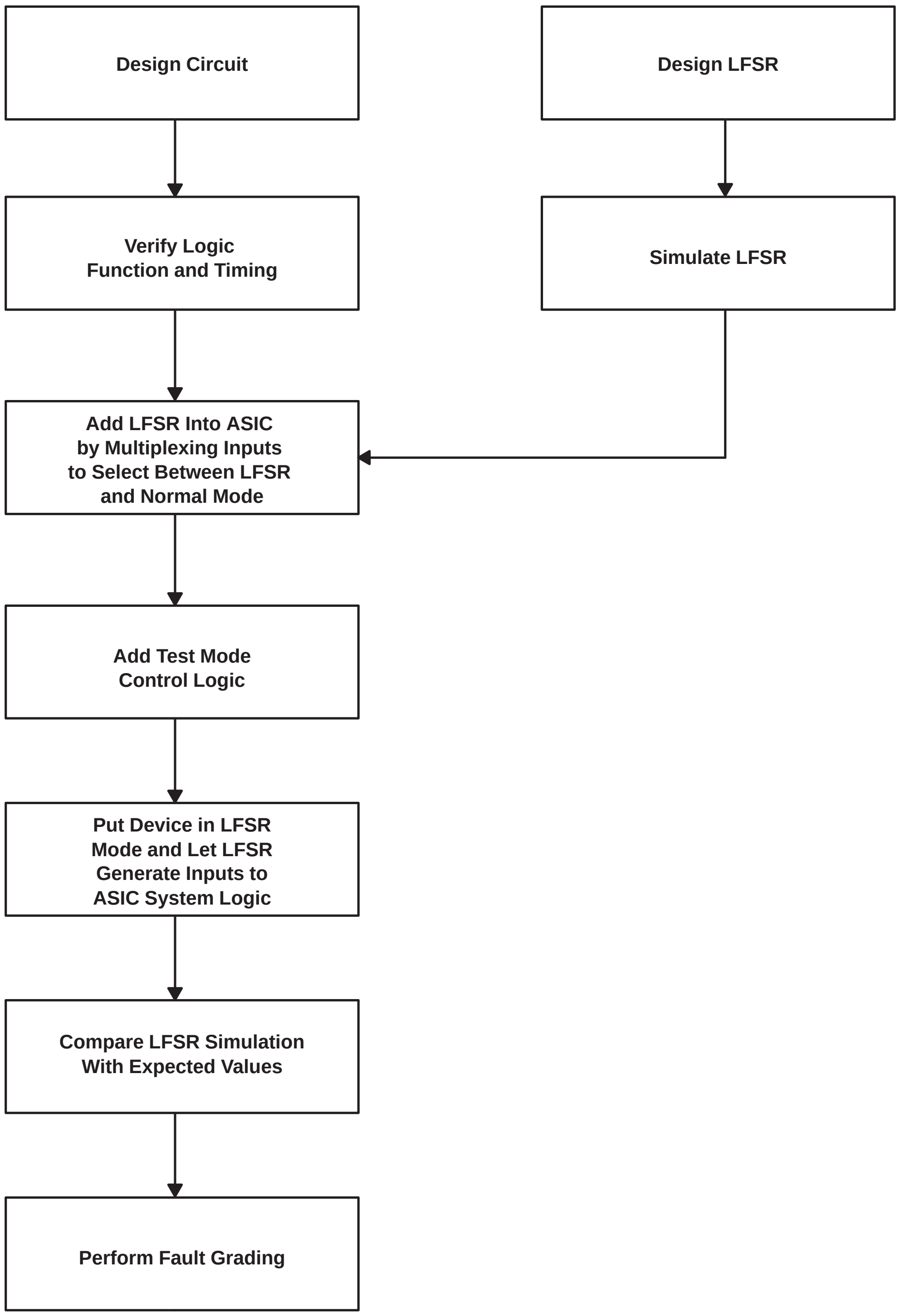
Design Circuit
Design LFSR
Verify Logic Function and Timing
Simulate LFSR
Add LFSR Into ASIC by Multiplexing Inputs to Select Between LFSR and Normal Mode
Add Test Mode Control Logic
Put Device in LFSR Mode and Let LFSR Generate Inputs to ASIC System Logic
Compare LFSR Simulation With Expected Values
Perform Fault Grading

Figure 4. Flowchart for Designing an LFSR Into an ASIC

External LFSRs

While the previous section focused on how to use the LFSR for built-in test (BIT) of an ASIC, it should be noted that the LFSR logic incorporated in an ASIC can also be used to drive external logic. This is especially easy to do if TI's SCOPE cells have been used to incorporate boundary scan into the design. (See the SCOPE and LFSRs section.) In this scenario, the LFSR would be multiplexed with the ASIC outputs so that when the device is placed in a test mode, pseudorandom patterns would be generated and applied to the board logic. Board faults could then be detected by monitoring the board's edge connector or by capturing the logic's outputs and scanning the information out through the IEEE 1149.1 test bus.

Pattern-Resistant Logic

A major potential problem with the LFSR approach is that some logic, such as an NA810 (an 8-input NAND gate), is pattern resistant. When any of the eight inputs is a 0, the output is a 1. Only when all eight inputs are 1s does the logic change state. A manufacturing defect, such as a stuck-at-0 fault on the output of the NA810, is extremely difficult to detect using random patterns. Another problem is that some control signals might not toggle as often as they would when stimulated by an LFSR. A system reset or a clear terminal is a good example. If this signal toggled all the time, the logic would keep resetting and poor fault detection might result. Sequential logic circuits also present a problem. For example, if state 3 in a state machine can only be achieved if the device first enters into state 1, then state 2, a random-pattern sequence may have difficulty in creating inputs that move the circuit from one state to another in the correct order.

These problems can be solved by a variety of techniques, including using longer LFSRs to generate virtually every possible input, holding key control signals to a fixed value during the test stage, and augmenting the patterns with hand-generated patterns to improve fault grading.

PSA

As discussed previously, an LFSR of any significant length will generate a large number of patterns. To avoid having to test the outputs of several-hundred thousand or more vectors, a parallel signal analyzer (PSA) is used to compress the data at the outputs of the ASIC. As Figure 5 indicates, the PSA is nothing more than an LFSR with exclusive-OR gates between the shift register elements. In fact, a PSA can be used as an LFSR if the A_IN, B_IN, and C_IN inputs are all held at 0. If the inputs are held at 0, the PSA will generate exactly the same patterns as the LFSR in Figure 2.

Historically, however, the PSA is most often used as a parallel-to-serial compression circuit. The A_IN, B_IN, etc. inputs are multiplexed with the ASIC outputs. As each pattern is applied to the ASIC by the LFSR connected to the ASIC inputs, the output state of the ASIC is read into the PSA. As each new pattern is applied, the PSA will perform an exclusive-OR of the last pattern's outputs with the current pattern's output to create a new value in the PSA. This, conceptually, is very similar to a calculator adding a series of numbers. For example, if adding 2 + 3 + 6 + 9 + 1 + 1 using a calculator, first add the first state, 2, to the second state, 3, to get 5. Then add the new state, 6, to the old state, 5, to get the new result 11, etc. Instead of using addition, the PSA performs an exclusive-OR of the series of 1s and 0s together to get the new result.

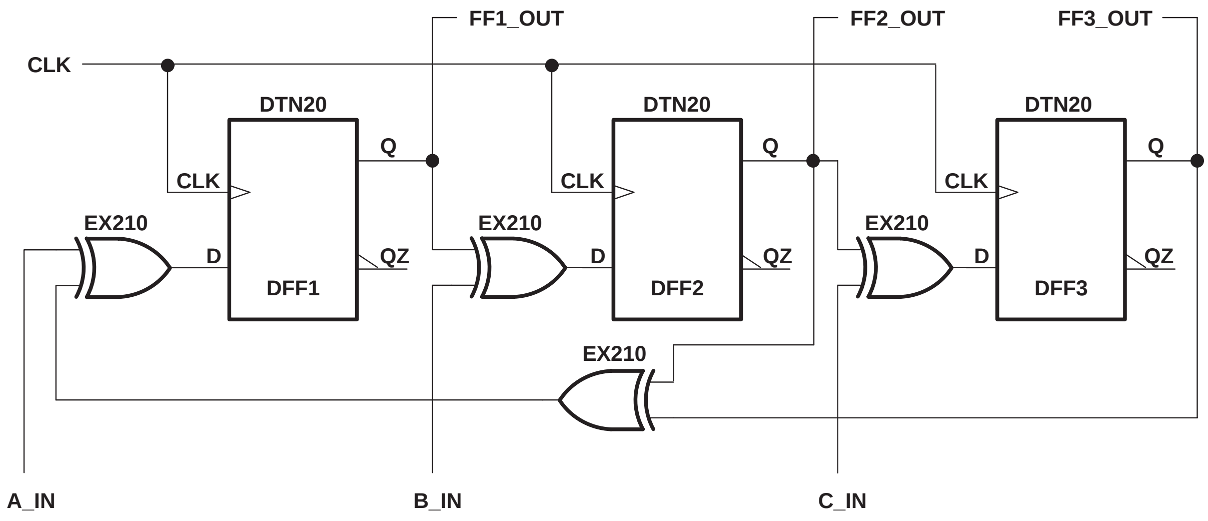

Figure 5. A Parallel Signal Analyzer

After a predefined number of patterns, the results in the PSA are read out of the ASIC (via FF1_OUT, FF2_OUT, and FF3_OUT) and compared to the expected value. Table 2 shows the signature that would be read out of the PSA if the inputs to the PSA were the outputs of the LFSR in Table 1.

Table 2. PSA Signatures

INPUTS					OUTPUTS		
CLOCK PULSE	A_IN	B_IN	C_IN		FF1_OUT	FF2_OUT	FF3_OUT
				SEED VALUE	1	0	0
1	1	1	1		1	0	1
2	0	1	1		1	0	1
3	0	0	1		1	1	1
4	1	0	0		1	1	1
5	0	1	0		0	0	1
6	1	0	1		0	0	1
7	1	1	0		0	1	0
8	1	1	1		0	1	0

Final signature = pattern 8, output results = 010

Aliasing

One major problem in compressing the results of a number of patterns into one pattern, called a signature, is aliasing. Aliasing occurs when the signature of the PSA is the same as expected, but the identity was caused by a cancellation of errors in the patterns. Thus, the silicon fault that caused the incorrect circuit response to the input pattern is never detected. To use the calculator example, 2 + 3 + 6 + 9 + 1 + 1 = 22, but so does 2 + 2 + 7 + 9 + 1 + 1. In this example, the second and third values that represent incorrect circuit values cancel each other out. Note that aliasing can occur only when there are two or more incorrect output patterns.

A way to minimize the aliasing problem is use maximal-length PSAs and to frequently read out the PSA signature for comparison to a known value or to monitor the most-significant PSA bit (FF3_OUT, in this case) continuously.

Summary

The LFSR is a shift register that has some of its outputs together in exclusive-OR configurations to form a feedback path. LFSRs are frequently used as pseudorandom pattern generators to generate a random number of 1s and 0s. Each output of the LFSR is multiplexed with an ASIC input and, when the device is placed in the LFSR (test) mode, the random, high-toggle-rate patterns produced are extremely good for generating high-fault coverage. To minimize the number of results that need to be compared to expected results, a PSA is used. The PSA compresses multiple parallel patterns into a single pattern signature that is compared to the expected value. If the signatures match, it is assumed that the ASIC passed the test vectors applied and there are no manufacturing defects.

Definitions

Sequential logic – logic functions whose next state depends on both the inputs to the function and its current state.

Combinatorial logic – logic functions that depend only on the inputs to the function.

Pattern-resistant logic – logic for which the pseudorandom pattern generation technique is not well suited because the random patterns will not cause a stuck-at fault to be caught.

Signature – the final result of the compression of a number of patterns using a PSA.

Aliasing – when the signature of a PSA is correct because two or more errors cancelled each other.

Stuck-at faults – manufacturing defects (such as a shorted transistor or an open metal line) that cause an input or an output of a logic function to be permanently stuck at a high or low logic level.

References

1 *Self-Test Services*, The STS ASIC Testability Seminar, Self-Test Services, Ambler, PA, 1989.

2 W.W. Peterson and E.J. Weldon, Jr. *Error Correcting Codes*, MIT press, Cambridge, MA 1972.

3 Texas Instruments, *SCOPE Cell Design Manual*, Texas Instruments Incorporated, Dallas, TX, 1990.

Acknowledgment

The author of this document is John Koeter.

(This page has been left blank intentionally.)