XAPP211 (v1.1) January 9, 2001

PN Generators Using the SRL Macro

Author: Andy Miller and Michael Gulotta

Summary

Pseudo-random Noise (PN) generators are at the heart of every spread spectrum system. Many PN generators are required within Code Division Multiple Access (CDMA) base stations. PN generators are used to implement synchronization and uniquely code individual user signals across the transmission interface. PN generators are based upon Linear Feedback Shift Registers (LFSRs). Every Look-Up-Table (LUT) in a Virtex™ series or Virtex-II series device can be configured as a 16-bit shift register (SRL16 macro). Hence, Virtex devices implement efficient LFSRs and deliver a significant reduction in resource utilization when compared with alternative flip-flop only PLD structures. For example, a 16-stage LFSR can be realized in just one LUT.

Introduction

Code Division Multiple Access (CDMA) systems are based upon several forms of spread spectrum techniques, the most popular being Direct Sequence Spread Spectrum (DS-SS).

Within a DS-SS system, the data being transmitted is spread across a wide radio spectrum using a pseudo random binary sequence unique to each user. Every data bit of a user signal is multiplied by many bits of a pseudo random binary sequence. This sequence is created by a PN generator and often referred to as a PN-Code.

The PN-Codes used within a CDMA system posses mathematical properties that enable them to coexist in the same spectrum with minimal interference. It is these properties that enable multiple users to exist in the same radio spectrum and hence leads to the term Multiple Access in CDMA.

It is the intention of this application note to discuss some of the terminology associated with PN Generators and demonstrate how they can be implemented in an area efficient manner using the Virtex/Virtex-II SRL16 macro.

PN Generators

A Pseudo-random Noise (PN) sequence/code is a binary sequence that exhibits randomness properties but has a finite length and is therefore deterministic.

There are three uses for PN sequences in DS-SS applications:

1. Spreading the bandwidth of the modulated signal over a wide radio spectrum.
2. Uniquely coding the different user signals that occupy the same transmission bandwidth in a multi-access system.
3. Synchronization for W-CDMA systems where there is no global timing reference.

In order to achieve these objectives, the coding sequences require special correlation properties referred to as auto correlation, and cross correlation.

Auto Correlation

Auto correlation is a measure of how well a signal *f* (t) can differentiate between itself and every time-shifted variant of itself.

Consider a data word of period seven bits at time $\delta t = 0$ (Table 1). If the 7-bit code were to be repeating within a discrete system then there are only six time-shifted replicas of the word, (shown in Table 1 for time $\delta t = 1$ to $\delta t = 6$). If each bit of the original ($\delta t = 0$), is compared with each bit of every time-shifted replica, then there are a number of agreements (A), and

disagreements (D), that when subtracted provide a measure of how closely the two words match (correlate).

In Table 1, the sequence "1110010" has a good auto correlation property as it provides a clear difference in the correlation value between itself and any time-shifted variant of itself.

In Table 2, the sequence "1111000" has the same number of bits, but the auto correlation property is not as good as there are some clear rejections of a match (correlation value = –5), and there are some "fuzzy" conditions where the time-shifted replica almost matches, (correlation value = 3).

Table 1: **Auto Correlation Example**

Sequence	Time Shift	(A)	(D)	(A–D)
1110010	δt = 0	7	0	7
0111001	δt = 1	3	4	–1
1011100	δt = 2	3	4	–1
0101110	δt = 3	3	4	–1
0010111	δt = 4	3	4	–1
1001011	δt = 5	3	4	–1
1100101	δt = 6	3	4	–1

Table 2: **Auto Correlation Example**

Sequence	Time Shift	(A)	(D)	(A–D)
1111000	δt = 0	7	0	7
0111100	δt = 1	5	2	3
0011110	δt = 2	3	4	–1
0001111	δt = 3	1	6	–5
1000111	δt = 4	1	6	–5
1100011	δt = 5	3	4	–1
1110001	δt = 6	5	2	3

Cross Correlation

Cross correlation is defined as the correlation between two different signals. Cross correlation is also calculated by subtracting the disagreements from the agreements, between two different sequences as opposed to the time-shifted replicas of the same signal.

Uniquely Coding the Different User Signals

In a CDMA system, each one of the multiple user signals in the receiver is assigned a unique PN code that behaves like a "key". From the examples in Table 1 and Table 2 it is evident that some sequences of the same length have better auto correlation properties than others and these special PN sequences are the ones used to code user signals in the system.

It is important to use a set of PN sequences that have a small cross correlation between each other in order to reduce an effect called adjacent channel interference. If the cross correlation between two PN sequences or "keys" is not small, there is a possibility that data coded from one user is incorrectly identified and assigned to another user because the two keys had a reasonable correlation. Research on small cross correlation PN sequences, have been carried out by many individuals but code sets identified by Kasami, R. Gold and Walsh are used throughout the IS-95 and UMTS W-CDMA systems.

When reviewing technical system specifications that require the use of PN generators or LFSRs, it is usual to find comments such as “codes from the Kasami set of Length…” or “Gold sequences generated with two polynomials of degree 41", or simply “x(n) = 1 + X^3 + X^7“.

The next section reviews some of the terminology associated with LFSRs in order to help match the system level definition to an architectural implementation.

LFSR Terminology

The heart of the PN generator is the LFSR. LFSRs sequence through (2^N – 1) states, where N is the number of registers in the LFSR. The contents of the registers are shifted right by one position at each clock cycle. The feedback from predefined registers or taps to the left most register are XOR-ed together.

LFSRs have several variables:

- The number of stages in the shift register.
- The number of taps in the feedback path.
- The position of each tap in the shift register stage.
- The initial starting condition of the shift register often referred to as the “FILL” state

Shift Register Length (N)

This is often referred to as the degree, and in general, the longer the shift register, the longer the duration of the PN sequence before it repeats. For a shift register of fixed length N, the number, and duration of the sequences that it can generate, are determined by the number, and position of taps used to generate the “parity” feedback bit.

Shift Register Taps

The combination of taps and their location is often referred to as a polynomial, and expressed as:

$$P(x) = 1 + X^3 + X^7$$

Where the leading “1” represents X^0, which is the output of the last stage of the shift register. X^3 is the output of register stage 3 and X^7 the output of XOR.

Various conventions are used to map the polynomial terms to register stages in the shift register implementation. The convention used in this application note is consistent with the convention used in the CDMA UMTS specification.

A couple of points to be noted about LFSRs and the polynomial used to describe them are:

- The last tap of the shift register is the leading “1” and always used in the shift register feedback path.
- The length of the shift register can be deduced from the exponent of the highest order term in the polynomial.
- The highest order term of the polynomial is the signal connecting the final “XOR” output to the shift register input. It does not feed back into the parity calculation along with the other taps identified in the polynomial.

Maximal Length Sequences (L)

A maximal length sequence for a shift register of length N is referred to as an m-sequence, and is defined as:

$$L = 2^N - 1$$

For example, an eight stage LFSR will have a set of m-sequences of length 255.

Correlation Properties

There are many combinations of taps that produce small cross correlation m-sequences. Consequently it is possible to define a set of taps that produce a collection of small cross correlation m-sequences for a constant length shift register. Kasami, Walsh, and Gold are recognized for the identification of small cross correlation maximal length PN codes.

The number of independent m-sequences (S), for a given length of shift register is defined by:

$$S \leq (L - 1) \div N$$

Gold Code Generator

The essence of a Gold Code generator (Figure 1) is that the outputs from two same-length LFSRs loaded with paired factor codes are Sword to create a new family of codes suited for use within CDMA systems. At the system level, a Gold Code generator is usually described by two polynomials that indicate the LFSR structure to be implemented.

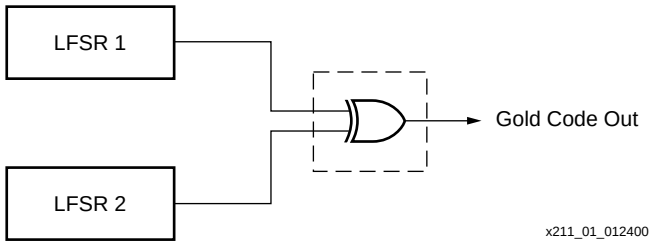

Figure 1: **Gold Code Generator**

The remainder of this application note considers the implementation of LFSRs in the Virtex series, Virtex-II series and Spartan-II family of FPGAs.

LFSRs Implemented in Virtex, Virtex-II and Spartan-II Devices

Each Virtex/Virtex-II LUT implements a 16-stage delay with the ability to connect any one of the 16 delay stages to the LUT output under control of the four address lines (A[3:0]). This mode of the LUT configuration can be accessed with an SRL16 primitive. In this mode, each LUT is effectively implementing a 16-stage shift register. This primitive will shift data on every clock cycle. A second primitive (SRL16E), provides the same shift register functionality with a shift register Clock Enable. Both the SRL16 and SRL16E implement area-efficient shift registers in one LUT with one data output pin in Virtex and Virtex-E devices (Figure 2).

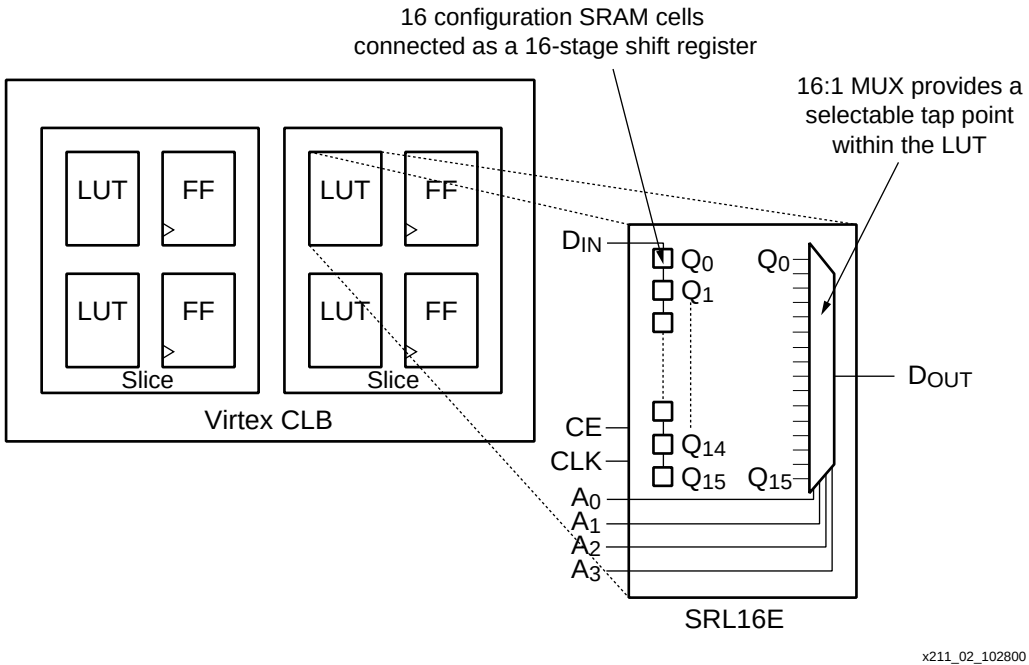

Figure 2: **SRL16E in Virtex, Virtex-E/EM Devices**

Figure 3 shows the implementation of SRL16E in a Virtex-II CLB.

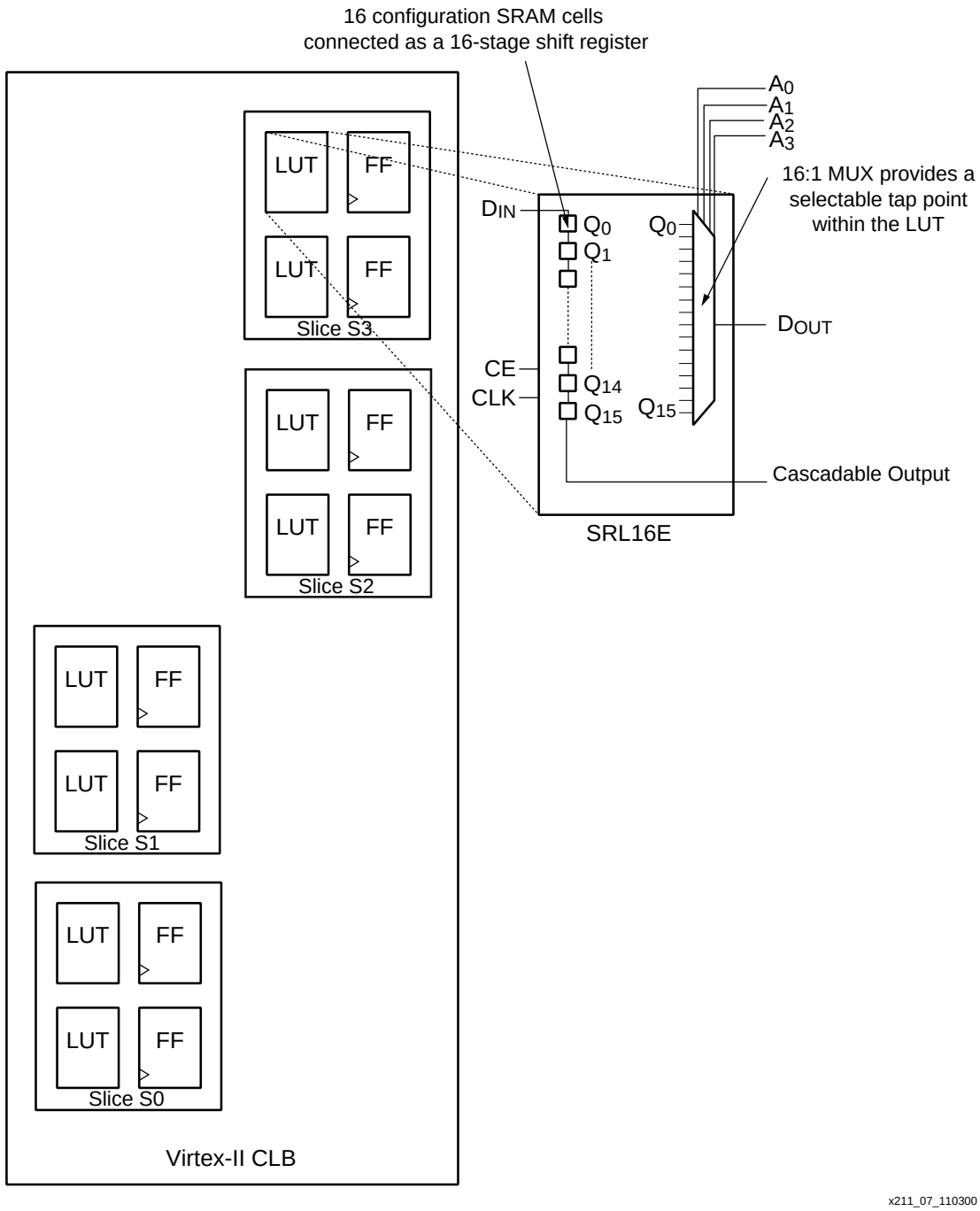

Figure 3: **SRL16E in a Virtex-II Device**

Multiple Shift Registers with Parallel Tap Access and Parity Calculation

A 16-stage LFSR with four selectable tap points can be designed with four SRL16 primitives as shown in Figure 4. An additional four input LUT is used to implement a parallel XOR parity calculation that is then fed back into the shift register as the new bit in the sequence. Tap D is the last stage in the shift register and so represents the LFSR output. The whole circuit is either clocked at a frequency known as the “chip rate”, or clocked by a higher frequency clock and enabled at the “chip rate”.

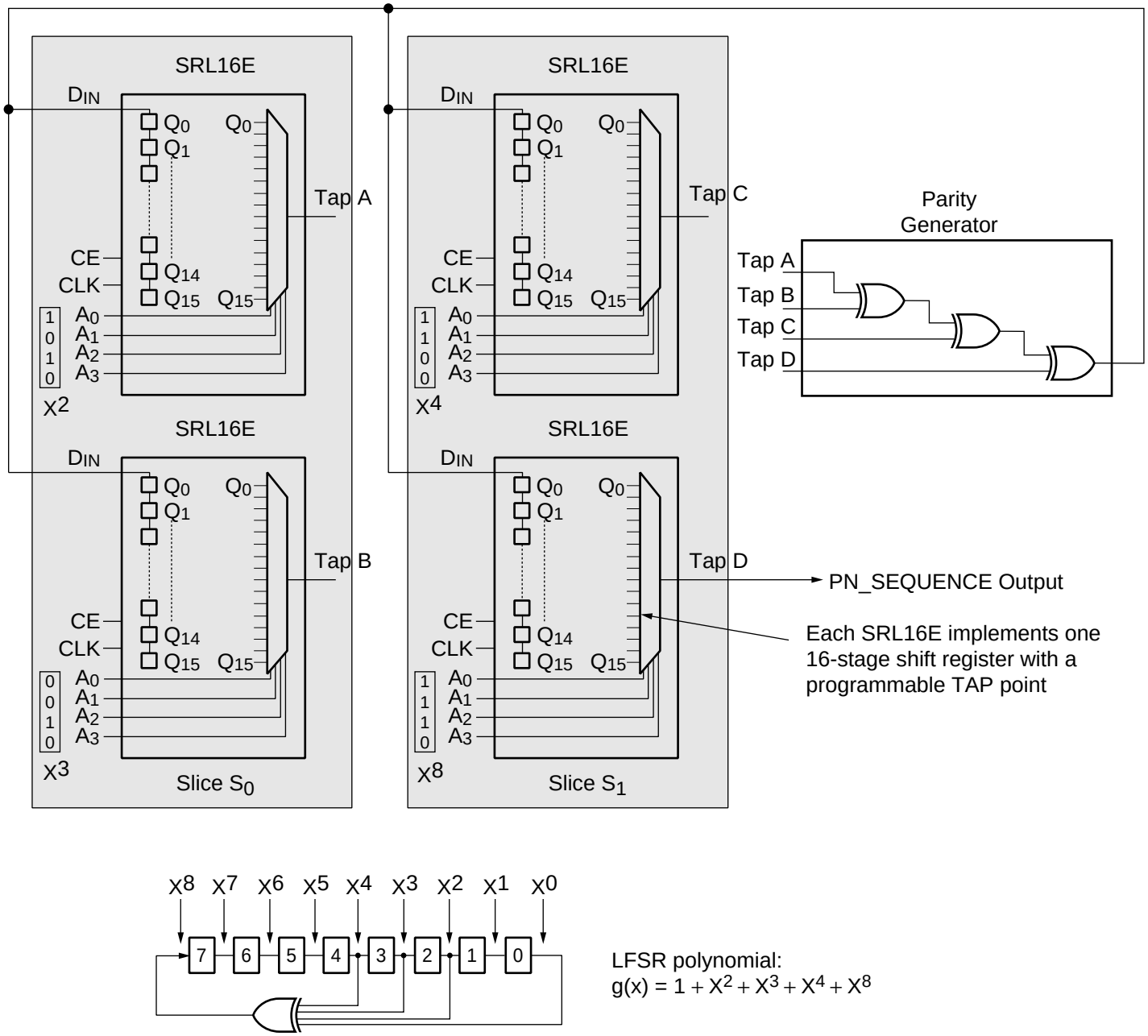

Figure 4: **Parallel 16-stage 4-tap LFSR**

PN Generator HDL Code

The Virtex/Virtex-II SRL16 macro can be inferred by using synthesis tools. This enables a significant area reduction to be achieved automatically using HDL code. As an example, the following HDL code (Table 3) will infer a 64-bit shift register using SRLs rather than FFs.

Table 3: **64-Bit SRL Shift Register Example**

VHDL	Verilog

process (clk)
begin
  if clk'event and clk='1' then
  Y <= Y(62 downto 0) & INPUT;
  end if;
end process
	Always @(posedge clk) begin
	  Y <= {Y[62:0], INPUT};
	End

Using SRLs instead of FFs, this circuit will cost only one Configurable Logic Block (CLB) instead of 16. With such dramatic savings, it is worth looking into ways to use SRLs whenever possible.

SRL16s implement registers using the configuration memory that have previously been considered an expensive overhead compared with ASICs. The user can now access those SRAM cells as part of the design through the use of LUT configuration modes such as RAM16x1 and the shift register modes SRL16 and SRL16E. The more that these elements can be bought into the functional operation of the design, the fewer number of dedicated flip-flop resources are required and so the device utilization can be reduced significantly.

The SRAM cells that form the SRL16 primitive are simple elements, however, there is no parallel access, nor can they be asynchronously reset. In a PN generator application it is often necessary to jump out of sequence requiring parallel loading of the LFSR with a predetermined state. This can still be satisfied with the SRL16 by serially filling the LFSR with a predetermined state. In Virtex-II, the16th output and one other output determined by the address line can be accessed simultaneously.

Filling the LFSR with an Initial Sequence

To achieve the LFSR serial fill, a mux is required in the LFSR feedback path allowing the loop to be broken while the predetermined state gets shifted in, as shown in Figure 5. The only caveat to implementing this pseudo-parallel fill with a serial fill is that the system must know in advance when the first bit of the new fill code is required to be output from the LFSR. The new serial fill sequence must be loaded into the LFSR over the same number of clock cycles that there are stages in the LFSR. The first bit of the new sequence must shift out of the LFSR on the first clock cycle after a parallel load would have occurred. The feedback path is reconnected and this first bit of the new sequence now contributes to the feedback bit entering the first stage of the LFSR.

The PN Generator in Figure 5 has 20 register stages. If a parallel load is required at time T = 0, then at time T = –20 clock cycles, the feedback path is broken when the signal Fill Sel is asserted. The shift register now contains the last 20 bits of the old sequence created before the feedback path was broken. It is unaffected by the feedback path now being disconnected from the shift register. Over the next 20 cycles from T = –20 to T = 0, the old sequence is shifted out, and the new fill sequence is shifted in. At T = 0, the new fill sequence has been shifted in, the feedback path is closed, and the next clock cycle delivers the first bit of the new sequence just as if the LFSR had been parallel loaded in a single cycle at T = 0.

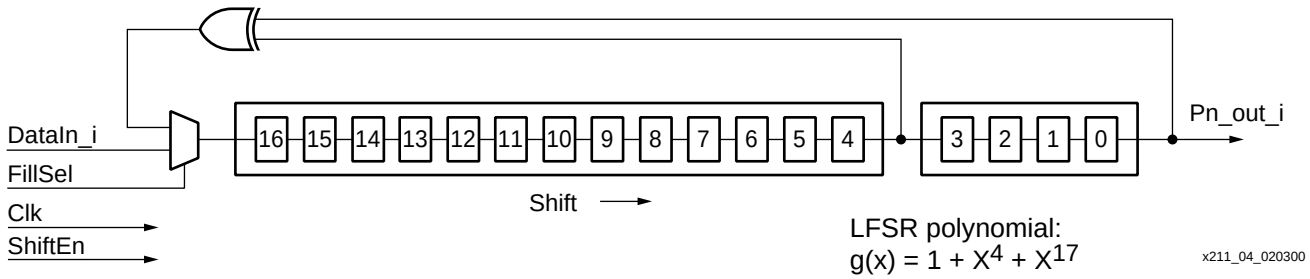

Figure 5: **PN Generator**

Puncturing and Augmenting

CDMA systems may require additional control of the basic LFSR. "Augmenting" the sequence with an additional state may be required to achieve 2^N states (instead of $2^N - 1$) in order to maintain an even modulo count. "Puncturing" the sequence by periodically skipping a state may be required if only a subset of the total 2^N states (e.g., 3 of 4) are required. Control signals have been provided in the HDL code to allow external circuitry to control such things as filling, puncturing, and stalling (augmentation).

LFSR Greater Than 16 Stages Long Using Multiple SRL16Es

Figure 6 demonstrates how an LFSR of length greater than 16 stages can be constructed from multiple SRL16s in Virtex and Virtex-E devices.

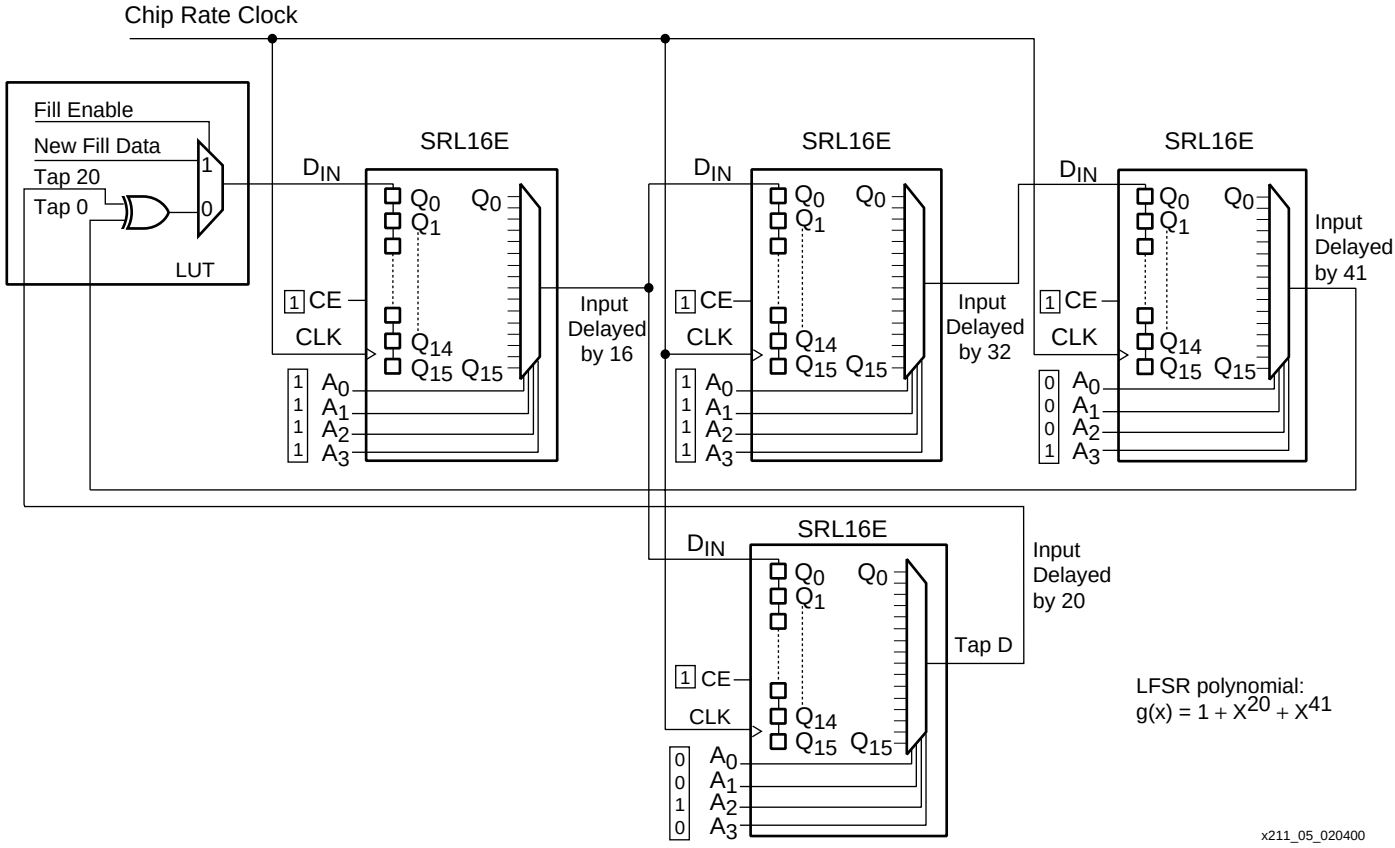

Figure 6: **41-stage, 2-tap LFSR with SRL16s in Virtex and Virtex-E Devices**

Figure 7 demonstrates how the same 41-stage, 2-tap LFSR can be implemented in a Virtex-II device using SRL16s. In Virtex-II devices, only three SRL16s are required due to the access to the 16th output in addition to the output selectable by the address lines.

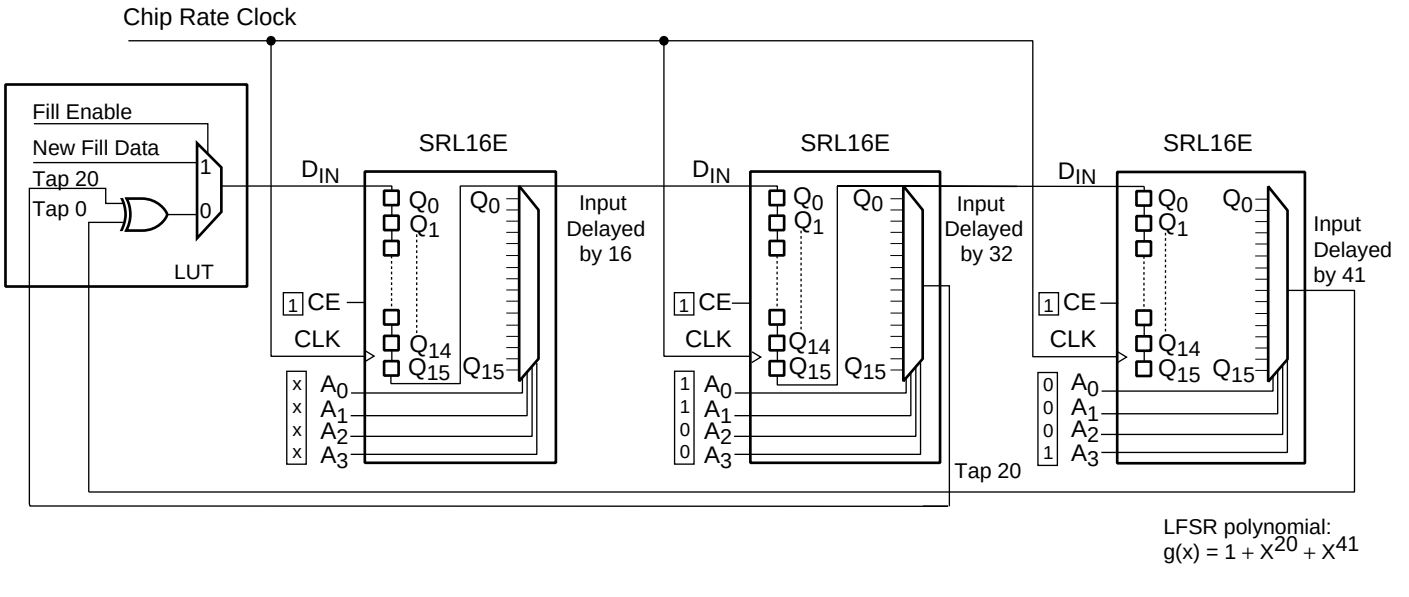

Figure 7: **41-stage, 2-tap LFSR with SRL16s in a Virtex-II Device**

HDL Code

Verilog and VHDL code examples have been written for the PN Generator module. The PN Generator provides two spreading sequences for the “I” (In-Phase) and “Q” (Quadrature phase) channels used in Quadrature Phase Shift Keying (QPSK) modulation schemes. The PN Generator HDL code therefore implements two LFSRs, one for the “I” channel and one for the “Q” channel.

In the verilog code, the number of LFSR taps are fixed, however the tap points and LFSR width are parameterizable. In the VHDL code, the number of taps, as well as, the tap points, and LFSR width are all parameterizable. The reference design is available on Xilinx’s ftp site at:

ftp://ftp.xilinx.com/pub/applications/xapp/xapp211.zip or **xapp211.tar.gz**

This code is not necessarily technology specific and can be used to target ASICs or FPGAs. When targeting Xilinx devices however, all the latest synthesis vendors (Leonardo, Synplicity, and FPGA Express) will infer the shift register LUTS (SRL16) resulting in a very efficient implementation. Table 4 is a comparison of performance and utilization results using all three synthesis tools.

Table 4: **Performance/Utilization of an XCV50 -6 Speed Grade Device**

	Design Implementation	Synopsys FPGA Express v3.3	Synplicity Synplify v5.2.2a	Exemplar Leonardo v1999.1g
Utilization	SRL16	6 slices	6 slices	6 slices
	Flip-Flops	24 slices	23 slices	24 slices
Performance	SRL16	178 MHz(2)	180 MHz(2)	220 MHz(1)

Notes:

1. Leonardo implements the design in just two levels of logic because the MUX, F5 in the Virtex slice, is inferred for the FillSelect MUX (shown in Figure 6).
2. Both FPGA Express and Synplify implement the design in 4 levels of logic because the MUX, F5 is not inferred for the FillSelect MUX (shown in Figure 6).

Conclusion

The Virtex and Virtex-II architectures are very efficient for creating PN generators by using the SRL. The SRL can also be used in many other applications such as pipe-line balancing, filters, dividers, and even waveform generators. In large systems, such as CDMA, the overall FPGA utilization can be reduced considerably by taking advantage of the SRL, which can lead to smaller, fewer, and less expensive parts. With only a basic understanding of the SRL, along with device friendly synthesis tools, these savings are easily accomplished without sacrificing code portability.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
02/04/00	1.0	Initial Xilinx release.
01/09/01	1.1	Addition of Virtex-II references.