Overview of Scrambler doc PhL 7.17.13

This has been updated and installed. It took a long time to get these summaries done!

Overview

This monograph purports to describe the theory underlying certain commonly used digital hardware circuits known as polynomial multipliers and dividers. A divider with no input is called a shift register generator, and one with input is known as a scrambler. This latter entity is usually restricted to the case in which the fixed divisor polynomial has a certain property: it is a "primitive polynomial" of a certain associated Galois Field.

The circuits involved are interesting in their own right and are not too hard to analyze in either the time domain or the frequency domain (the Z transform domain). Far more interesting is the connection between these circuits and the subject of finite fields known as Galois Fields in honor of Évariste Galois. The Galois Field connection allows certain conclusions to be reached concerning the behavior of polynomial processors which would be very difficult to ascertain by other means. Unfortunately, the subject of Galois Fields is somewhat obscure and often does not appear in the main line curriculum of the undergraduate student in engineering, communications or computer science, though it does appear wherever cyclic codes are discussed. As an aid to the reader, the author has written a separate set of notes on this subject which is referred to in this document as GA (see References).

Like the rodeo cowgirl who stands on two horses as they prance around a corral, the topics addressed in the current document stand upon two mathematical pillars. A student of these topics no doubt experiences the same wobbly uncertainty felt by the cowgirl and it is not hard to fall off and give up. The first pillar is the subject of Galois Fields already mentioned. The second pillar is the whole universe of topics associated with the Fourier Transform and its digital descendents such as the Z Transform. The whole concept of a polynomial processor is that it processes polynomials in the variable z of the Z transform. The circuits are simple to analyze in this "z domain", but in the real world we are also interested in time-domain behavior, and these worlds are connected by the Convolution Theorem of Fourier theory. Another connection is that one is often interested in the frequency spectrum and the spectral power density of the output signal of a shift register generator or a scrambler, and just understanding the origin of spectral power density formulas is a whole separate effort. In a separate document referred to as FT the author has provided some notes on these matters, again see References.

A smaller but no less painful third mathematical pillar involved in this document involves probability, statistics and random variables. These topics are addressed in Appendix G of FT.

Summary

There is a lot going on in each of the three chapters of this document, and the overviews below are correspondingly fairly detailed. A briefer overview is provided by the Table of Contents.

Chapter 1 : Polynomial Processors

Chapter 1 opens by showing the four basic hardware polynomial processors considered in this document. The Type A divider and multiplier have outboard adders, while the Type B have inboard adders. The

input and output symbol streams are serial streams. More costly parallel implementations, though certainly possible, are not treated.

Section 1.1 notes that any "polynomial" F(z) is the Z Transform of a sequence f_n, so the latter is the **time-domain** representation of a signal while the polynomial is the **z-domain** representation. Certain rules are noted, such as $f_n \leftrightarrow F(z) \Leftrightarrow f_{n-m} \leftrightarrow z^{-m} F(z)$.

DIVIDERS

In **Section 1.2** the **Type A** divider is drawn in Fig 1.1 and the lines are interpreted as carrying **symbols** which could be bits or bytes or something else. In most of the paper, these symbols are treated as elements of the **Galois Field GF(p)** rather than $GF(p^m)$. By starting with a simple time-domain description of the divider, the fact that O(z) = I(z)/H(z) is quickly obtained, verifying that Fig 1.1 in fact implements a polynomial divider. Here I(z) is a possibly infinite polynomial whose coefficients are the symbols i_n which are clocked into Fig 1.1, while O(z) is the normally infinite polynomial whose coefficients are the symbols o_n which are clocked out. H(z) is a polynomial of fixed coefficients h_i which are part of the Fig 1.1 circuit. A distinction is made between **proper** polynomials which have non-negative powers only, and **improper** polynomials which can include negative integer powers. Whereas the divisor H(z) is a proper polynomial of degree k, I(z) and O(z) are improper polynomials each of which in general has an infinite number of terms. Looking at the "quotient" O(z) and its infinite number of terms, one wonders "where is the remainder?" It is seen that if improper polynomials are allowed, the notion of a "remainder" is completely arbitrary and depends on "where you stop" in the division process. Examples of this situation are presented.

Section 1.3 compares the infinite sequence of the output of a polynomial division to the infinite repeating sequence of digits of the division of two real numbers. The output of a polynomial divider is then considered after a fixed number r of input symbols are processed. In this case, the division process can be thought of as $O_{r-k}(z) = I_r(z)/H(z)$ where $I_r(z)$ and H(z) are "proper", but $O_{r-k}(z)$ is still an improper polynomial. One can then write $O_{r-k}(z) = \mathcal{Q}(z) + R(z)/H(z)$ where the quotient $\mathcal{Q}(z)$ is a proper polynomial of degree r-k and the remainder R(z) is a proper polynomial of degree < k. At this point, if another input symbol is processed, the interpretation changes to accommodate r+1 input bits, and we have a whole new division problem to interpret. In general we find $I_r(z)/H(z) = \mathcal{Q}^{(r)}(z) + R^{(r)}(z)/H(z)$ so as the bits come in, we get in fact a sequence of quotients and a **sequence of remainders**. Even if the input sequence vanishes after some point, these sequences continue on in general forever. One is normally used to a division having a unique quotient and remainder, but as the input bits come in, the division problem keeps changing and so too do the quotient and remainder. It turns out that in the Type A divider, the remainder is not "visible" and in fact is a function of both the register state and some symbols already shifted out as coefficients of O(z). Finally, it is noted that 1/H(z) is the **impulse response** of the divider, since in this case I(z) = 1, meaning the divider is pulsed by a single unity symbol at time t = 0. If the divider were interpreted as a filter, 1/H(z) would be that filter's **transfer function**.

Section 1.4 repeats the above analysis for the **Type B** divider of Fig 1.5 where the adders are "inboard". One again finds that O(z) = I(z)/H(z).

Section 1.5 studies *how* the two kinds of dividers actually "work". The action of each circuit on each clock is related to the process of "grade school division" of polynomials carried out just as one divides numbers by "long division". In either Type A or Type B, there is a "current dividend" at each step. It is seen that for the Type B divider, the sequence of current dividends is in fact the sequence of remainders discussed just above. For this kind of divider, the remainder is then "visible" and is precisely the state of the registers.

MULTIPLIERS

Section 1.6 addresses the **Type A** polynomial multiplier, Fig 1.8. It is shown that $z^k\ O(z) = H(z)\ I(z)$, so apart from the factor z^k, the circuit really does multiply two polynomials. An interpretation of the form $O_{r+k}(z) = H(z)\ I_r(z)$ is then provided in which all three polynomials are "proper". The impulse response is $z^k\ O(z) = H(z)$. The factor z^k is associated with a simple time shift of the output by k clocks.

Section 1.7 then treats the **Type B** multiplier, and $z^k\ O(z) = H(z)\ I(z)$ is again obtained, as is the same proper polynomial interpretation $O_{r+k}(z) = H(z)\ I_r(z)$ and the same impulse response $z^k\ O(z) = H(z)$.

Section 1.8 shows how these multipliers "work" in terms again of "grade school multiplication" of polynomials. For both the dividers and multipliers, we see how the inboard and outboard adder topology affects the method by which contributions are added up to get the desired output. Up to this point, almost everything has been in the z-domain, since this is where the polynomials of our polynomial processors live.

In **Section 1.9**, the circuits are analyzed instead in the **time domain**. The Type A multiplier is treated first, giving a time-domain equation $o_{k+n} = \Sigma_{j=0}^{k}\ h_j i_{n+j}$. The Type A divider is done next, with result $i_n = \Sigma_{j=0}^{k}\ h_j o_{n+j}$ which is a difference equation for the output o_k. Section (c) shows how the Type A time-domain and z-domain equations are consistent with a Z Transform theorem known as **The Convolution Theorem**. Since the Type A and Type B z-domain equations are the same, it is concluded from this theorem that the Type B time-domain results must be the same as for the Type A, so there is no reason to separately compute the Type B time-domain equations.

At this point the basic results are summarized in box (1.9.8).

BOTH AT ONCE

Section 1.10 the considers a circuit Fig 1.13 which does *simultaneous* multiplication of an input polynomial I(z) by polynomial H(z) and division by another polynomial G(z). The result of this combined operation is O(z) = I(z)H(z)/G(z). It is shown how the limits H(z) = 1 and $G(z) = z^k$ reproduce the results of the separate divider and multiplier circuits treated earlier.

Section 1.11 shows how the combined circuit of the previous section is used in **CRC** error detection.

Chapter 2: Shift Register Generators

Section 2.1 defines a state machine and its state vector period N. A shift register generator is defined to be either the Type A or Type B polynomial divider studied in Chapter 1 in which the input stream is set to 0. A shift register generator is an example of a state machine with no inputs and one output. It has k registers each of which holds a symbol of GF(p^m), but in the next section m is restricted to m = 1.

Section 2.2 studies the evolution of the state vector of a Type B shift register generator and seeks to learn about the nature of the state vector period.

(a) The k register values (the state) of a Type B shift register generator are encoded as coefficients of a polynomial q(x), and it is shown how the state updates over one clock period: $q'(x) = x\,q(x) - o\,h(x) + i$.

(b) Our interest from this point on is limited to registers which contain GF(p) symbols.

(c) Certain facts about **Galois Fields** GF(p^k) are reviewed.

(d) The initial encoded state vector q(x) is identified with an element β_0 of GF(p^k) and after evolving for s clocks it is shown that $\beta_s = \alpha^s \beta_0$ (the Galois Iterator), where α is a different element of GF(p^k). It is shown that if h(x) is chosen to be a primitive polynomial of GF(p^k), then the state vector period is the maximal possible value $p^k - 1$.

(e) The state vector after s clocks is identified with a polynomial division remainder after s clocks. The period n of h(x) is defined as the smallest integer n for which $(x^n-1)/h(x)$ is a polynomial. It is shown using the remainder idea that the Type B state vector period N is equal to this period n of h(x).

Section 2.3 studies the output and the output period of a shift register generator of either type. It is shown that the output sequence $\{o_j\}$ of such a generator satisfies the time-domain equation $\Sigma_{j=0}^{k} h_j o_{n+j} = 0$ for n = 0,1.2... . The solution sequence $\{o_j\}$ of this equation is shown to have interesting properties. There are exactly p^k-1 non-zero solutions, and all solutions can be written as $\{o_i\}$ = {**c**,**c**,**c** ...} where **c** is an n-symbol code word of the **cyclic code** generated by $g(x) = (x^n-1)/h(x)$ where n is the period of h(x). Since cyclic code words have simple properties (any rotation of **c** is another code word and the sum of two code words is a code word), it is easy to find the exact solutions $\{o_i\}$. The case GF(2^4) is treated as a detailed example. It is shown that, if h(x) is a primitive polynomial, then n = p^k-1 and all of the p^k-1 solutions $\{o_j\}$ have the same period P = p^k-1, and that in fact all solutions are just time-shifts of a single sequence which is called a **characteristic sequence** or **maximal length sequence** (MLS). The 511 bit MLS sequence is displayed for GF(2^9). For a Type A generator, it is shown that the output period P is the same as the state vector period N. The notion of a sift list associated with an MLS sequence is defined and it is shown that in one period of an MLS sequence, each symbol appears k times in the associated sift list.

Section 2.4 studies the properties of an MLS sequence produced by any shift register generator whose h(x) is a primitive polynomial of GF(p^k). Any particular k-long sequence can start only once in an MLS period. The largest run of zeros is length k-1, while that of any non-zero symbol is k, and all these runs occur only once per MLS period. In an MLS sequence, 0 occurs p^{k-1}-1 times while each non-zero symbol occurs p^{k-1} times. An MLS sequence differs from a non-trivial shifted version of itself in $(p-1)p^{k-1}$ places. In a small table (2.4.8), the sum and product of an MLS sequence with a shifted version of itself are studied. It is claimed without proof (which comes in the next section) that for reasonably large k, the binary MLS sequence is very close to a white sequence for which the probability of finding runs of length n of either symbol is given by $P(n) = 1/2^n$. The error between a white sequence and an MLS sequence is estimated.

Section 2.5

(a) States the spectral power density of a **statistical pulse train** whose amplitudes form a white sequence. This result is quoted from FT for a general pulse shape and is then specialized to the box pulse shape. The spectrum is stated for symbols in both {1,0} and {1,-1}.

(b) defines the **autocorrelation** sequence r_s for a set of pulse train amplitudes y_n and then quotes a theorem from FT concerning infinite sequences composed of a repeating subsequence of length P. The theorem states the corresponding pulse train's spectral power density provided certain conditions are met concerning r_s.

(c) computes the autocorrelation sequence r_s for an MLS sequence with symbols in {1,0}. It is shown that this r_s meets the conditions of the theorem of the previous section, and then the spectral power density is stated for an MLS sequence, first for a general pulse shape and then for the box pulse shape. Since the pulse train amplitude sequence is periodic (with period P), the spectrum is entirely discrete and has an interesting structure which is plotted in Fig 2.8. As $P \to \infty$, the spectral lines (apart from the DC line) coalesce into a continuum which is the white sequence spectrum.

(d) repeats the previous section for symbols in {1,-1} and then compares the results of the two sections.

(e) displays a table summarizing the statistics for uncorrelated, MLS and white sequences for both {1,0} and {1,-1} cases, and then calculates the MLS correlation in each case, showing explicitly how MLS sequences are in fact *not* uncorrelated. Finally, the autocorrelation sequence for each case is plotted.

Chapter 3: The Matrix Approach to Scramblers

Section 3.1 reviews the developments of previous sections.

Section 3.2 then develops a linear-algebra style solution for the state vector of a Type B divider given an arbitrary initial state vector and an arbitrary input sequence. The solution involves powers of the kxk companion **matrix** B associated with the divider's primitive polynomial h(x). A connection is made between this solution and a solution previously obtained in Section 2.2 in terms of abstract GF(p^k) Galois Field elements. The state vector has k elements in GF(p) and can be interpreted as a GF(p^k) field element.

Section 3.3 finds the corresponding solution of the Type A polynomial divider using an alternative companion matrix A.

Section 3.4 revisits the topic of shift register generators broached in Chapter 2 and develops some new results based on the matrix formalism of the previous sections. It is found that each register of either type of shift register generator cycles through the MLS sequence if h(x) is a primitive polynomial of GF(p^k).

Section 3.5 formally defines a **scrambler**, then comments on its relation to **spread spectrum** communication. It is shown that each matrix element of a matrix representation of a Galois Field based on some primitive field element cycles through the MLS sequence described in Chapter 2. The remainder of the section describes how a scrambler "whitens" the statistics of its input stream.

Section 3.6 then studies the **spectral power density** of a scrambler output in the limit of a very long MLS period. The power density of the scrambler output is "white" and effectively continuous, and this is compared to the discrete spectral power density of a simple pulse train having the same pulse period T_1.

Section 3.7 discusses the NRZI **mini-scrambler** which is a polynomial divider with a single register. This scrambler when appended to a longer one produces an output signal which is non-polar. This section then ends with a summary of scrambler results.

Section 3.8 mimics Section 3.3, finding a linear-algebra solution for the state vector of a Type A polynomial *multiplier* (Section 3.3 did the divider). A new matrix D appears. As a check, the output sequence of the multiplier is computed in this matrix formalism and agrees with a result found earlier.

Section 3.9 repeats this process for the Type B multiplier and the same matrix D appears.

Section 3.10 makes use of the matrix solution found in Section 3.3 for the Type A divider (scrambler) and shows that a **descrambler** really does descramble the output of a scrambler. This fact is trivial to show in the z domain, but is not so easy to show directly in the time domain. This fact is then applied to a standard form of scrambling involving both a "main scrambler" and the NRZI "mini-scrambler".

Section 3.11 notes that a mini-scrambler passes through strings of zeros, and one is then concerned with the length of strings of zeros that can be created by the main scrambler. The concern is that long strings of zeros stress a receiver's **clock recovery** circuitry and can possibly throw it out of lock resulting in data loss. The worse-case situation occurs when the scrambler's input stream forms a **kill vector** for the current scrambler state, throwing the scrambler into the all-zeros state. Examples of strings of zeros are considered in some serial digital video standards. For a given scrambler state, the kill vector is calculated using simple matrix methods. The section concludes with comments on the effect of long strings of zeros.

Appendix A explores an analogy between real numbers expressed in decimal notation and polynomials whose coefficients lie in Mod(10) = Z_{10}.

Appendix B proves a certain claim made in (2.3.8) concerning the output sequence of a shift register generator.

Appendix C presents E.J. Watson's list of primitive polynomials for GF(2^k).

Appendix D proves a certain sum rule (3.10.15) involving the companion matrix A and the coefficients of the divisor polynomial h(x).