Selection of Polynomials for Cyclic Redundancy Check for the use of High Speed Embedded – An Algorithmic Procedure

A. Ahmad and L. Hayat
Department of Electrical and Computer Engineering,
College of Engineering, Sultan Qaboos University
P. O. Box 33, Postal Code 123; Muscat, Sultanate of Oman
Tel.: (968) 24415327, Fax. (968) 24413454
E-mails: afaq@squ.edu.om

Abstract: - Cyclic Redundancy Check (CRC) technique which is widely used tools in globally standardized telecommunications systems for dealing with data errors detection and correction have not been fully standardized. Most of the CRCs in current use have some weakness with respect to strength or construction. Standardization of CRCs would allow for better designed CRCs to come into common use is primarily limited due to the complexity of search procedures of the primitive characteristic polynomials. To this direction this paper proposes a method of simplifying the computation and complexity of the search procedure of the primitive characteristic polynomials in order to facilitate implementation of the circuitry for high-speed CRC computation in standard CMOS technology.

Key-Words: - Cyclic Redundancy Check, CRC, Linear Feedback Shift Registers, LFSRs, Primitive Polynomial Primitive Characteristic Polynomial, Power Dissipation, Exclusive-OR, D-Flip-Flop

1 Introduction

A CRC is an error-detecting code. The CRC methodology is based on cyclic algorithm that generates redundant information. The CRC technique is heavily used in data communication, storage devices and VLSI testing. Linear Feedback Shift Registers (LFSRs) along with Exclusive-OR logic gates is the possible way out to realize the hardware solutions for CRC. While on the other hand the simulation of polynomial division process realizes the CRC in software manner. In general, LFSR with serial data feed has been used to implement the CRC algorithm. This LFSR based method simply performs a division and then the remainder which is the resulting CRC checksum, is stored in the registers after each clock cycle of the transmitted bit, and at the end of the last bit entry of the data records the final check sum of the transmitted data [1] – [4]. An n-bit LFSR based CRC implementation is as shown in Figure 1.

In general an n-bit LFSR based CRC circuit may give 2^{n-1} different characteristic polynomials. For example a 3-bit CRC circuit may have any of the 4 possible characteristic polynomials, namely these are $1+x^3$, $1+x^2+x^3$, $1+x+x^3$, and $1+x+x^2+x^3$. Table 1 provides some of the used characteristic polynomials. The use of the different characteristic polynomials results in different efficiency in terms of error masking capability along with hardware, time, and power requirements. As it has been claimed that the use of primitive characteristic polynomials are better for the practice but careful investigations of primitive polynomials are required. However, finding an optimal solution of selecting appropriate characteristic polynomials is restricted due to the search problem of primitive characteristic polynomials [5] – [11]. Figure 2 maps the complexity of the search problem of primitive characteristic polynomials of order n = 1 to 24. In Figure 2, NP represents the total number of possible characteristic polynomials whereas; NPP indicates the total number of possible characteristic primitive characteristic polynomials of order n. The complexity of the search problem increases as the value of n increased. Due to this problem many of the CRC characteristic polynomials which are in use are not the primitive one. Refer to the Table 1, where it can be verified that the characteristic polynomials of the circuits CRC-32K (Koopman), CRC-32C (Castagnoli) are not primitive characteristic polynomials. Hence this paper extends the research work of finding the primitive characteristic polynomials in an efficient manner.

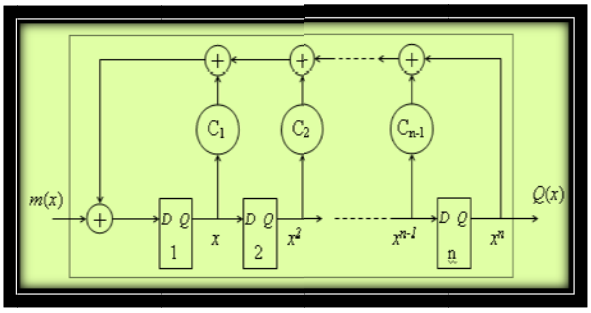

Fig. 1: A general LFSR based CRC circuit

Table 1: Different characteristic Polynomials Usage

CRC-4-ITU	x4 + x + 1 (ITU G.704)
CRC-5-ITU	x5 + x4 + x2 + 1 (ITU G.704)
CRC-5-USB	x5 + x2 + 1 (USB token packets)
CRC-6-ITU	x6 + x + 1 (ITU G.704)
CRC-7	x7 + x3 + 1 (telecom systems, MMC)
CRC-8-ATM	x8 + x2 + x + 1 (ATM HEC)
CRC-8-CCITT	x8 + x7 + x3 + x2 + 1 (1-Wire bus)
CRC-8-Dallas/Maxim	x8 + x5 + x4 + 1 (1-Wire bus)
CRC-8	x8 + x7 + x6 + x4 + x2 + 1
CRC-8-SAE J1850	x8 + x4 + x3 + x2 + 1
CRC-10	x10 + x9 + x5 + x4 + x + 1
CRC-12	x12 + x11 + x3 + x2 + x + 1 (telecom systems)
CRC-15-CAN	x15 + x14 + x10 + x8 + x7 + x4 + x3 + 1
CRC-16-CCITT	x16 + x12 + x5 + 1 (X.25, V.41, Bluetooth, PPP, IrDA, BACnet; known as "CRC-CCITT")
CRC-16-IBM	x16 + x15 + x2 + 1 (XMODEM, USB, many others; also known as "CRC-16")
CRC-24-Radix-64	x24 + x23 + x18 + x17 + x14 + x11 + x10 + x7 + x6 + x5 + x4 + x3 + x + 1
CRC-32-MPEG2	x32 + x26 + x23 + x22 + x16 + x12 + x11 + x10 + x8 + x7 + x5 + x4 + x2 + x + 1
CRC-32-IEEE 802.3	x32 + x26 + x23 + x22 + x16 + x12 + x11 + x10 + x8 + x7 + x5 + x4 + x2 + x + 1 (V.42)
CRC-32C (Castagnoli)	x32 + x28 + x27 + x26 + x25 + x23 + x22 + x20 + x19 + x18 + x14 + x13 + x11 + x10 + x9 + x8 + x6 + 1
CRC-32K (Koopman)	x32 + x30 + x29 + x28 + x26 + x20 + x19 + x17 + x16 + x15 + x11 + x10 + x7 + x6 + x4 + x2 + x + 1

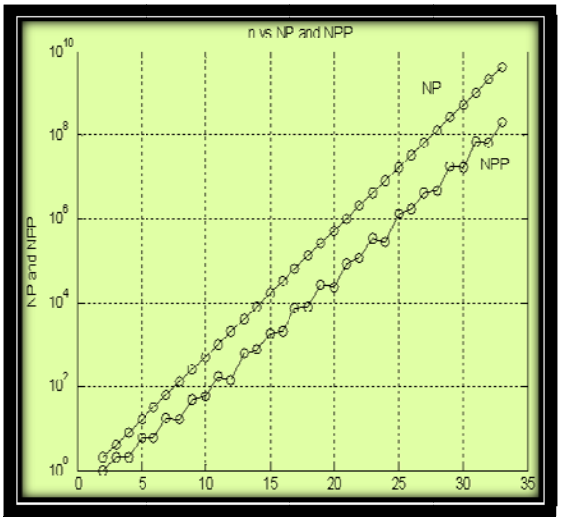

Fig. 2: The values of n vs NP / NPP

2 Primitive Polynomials - Search

This section briefly presents the mathematical background LFSR.

Theorem 1: A primitive polynomial is irreducible if the smallest positive where n is the degree of the polynomial.

Theorem 2: A polynomial P(x) of degree n is primitive, if and only if, its reciprocal polynomial $P^*(x)$ is also primitive.

Let A represent the state transition matrix of order n×n, for an n-stage LFSR. Let the state of the LFSR at any time 't' be represented by vector $[Y(t)]=[y_1(t),y_2(t), ...,y_n(t)]$

$$\begin{bmatrix} y_1(t+1) \\ y_1(t+1) \\ \vdots \\ y_{n-1}(t+1) \\ y_n(t+1) \end{bmatrix} = \begin{bmatrix} c_1 & c_2 & c_3 & \cdots & c_{n-1} & c_n \\ 1 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 1 & 0 \end{bmatrix} * \begin{bmatrix} y_1(t) \\ y_1(t) \\ \vdots \\ y_{n-1}(t) \\ y_n(t) \end{bmatrix} \quad (1)$$

Where $\mathbf{c_j = 0 \text{ or } 1}$, for $1 \le j \le n-1$
$\mathbf{c_j = 1}$, for $j = n$. (2)

Equation (1) can be written as

$$[Y(t+1)] = [A][Y(t)] \quad (3)$$

On the basis of the property of periodicity of LFSR and Equation (3), it follows that

$$[Y(t)] = [Y(t+p)] = [A]^p[Y(t)] \quad (4)$$

Assertion 1:
The period p of an n-bit LFSR will only be maximal when $p = m = 2^n-1$.

Definition 1:
For the matrix A of LFSR, the characteristic polynomial is given by

$$\varphi(x) = 1 + \sum_{j=1}^{n} c_j x^j \qquad \mathbf{c_n=1.} \quad (5)$$

Definition 2:
A characteristic polynomial P(x) associated with a maximal length sequence is called a 'primitive polynomial'.

Theorem 3:
If a primitive characteristic polynomial P(x) of an n-stage LFSR has corresponding feedback connections from stages n, k, h, ..., then the reciprocal of this polynomial $P^*(x)$ will have corresponding feedback connections from stages n, n-k, n-h, ..., etc. (n>k>h..).

Theorem 4:
If the characteristic polynomial of an n-stage LFSR is irreducible, then the period of generated sequence by the LFSR is a factor of 2^n-1.

Corollary 1:
If $p = m = 2^n$-1 is a prime number, then the characteristic polynomial corresponds to that connection of LFSR will be primitive, if and only if $[A]^m = I$.

Corollary 2:
If $p = m = 2^n$-1 is not a prime number, and $[A]^p i = I$, where p_i is a divisor of p, then the characteristic polynomial corresponds to that connection of LFSR can not be primitive.

Theorem 5:
The characteristic polynomial of an n-stage LFSR can be primitive, if and only if the number of tapping in that LFSR is even.

The determination of the primitive polynomial involves the use of the Euler phi-function Φ(.) and search for primes.

Theorem 6:
If m is a composite integer, then m has a prime factor not exceeding √m.

Lemma 1:
Every positive integer grater than one has a prime divisor.

Theorem 7:

Let $m = p_1^{a_1} p_2^{a_2} \cdots p_i^{a_i} \cdots p_k^{a_k}$ be the prime-power factorization of the positive integer m.
Then

$$\Phi(m) = m\left(1-\frac{1}{p_1}\right)\left(1-\frac{1}{p_2}\right)\cdots\left(1-\frac{1}{p_k}\right) \quad (6)$$

Theorem 8:
The total number of possible primitive polynomials (NPP) of order n is given by

$$NPP = \frac{\Phi(m)}{n} \quad (7)$$

3 Algorithm Design

Based on our rigorous investigations we devised efficient algorithms which are presented below.

3.1 Algorithm:

3.1.1 Computing Prime Factors of p

Input: n; Output: p, prime factors of p (p_i), number of prime factors (k), and exponents of each prime factor (e_i).

1. Read n and do the followings
2. Compute $p = 2^n - 1$;
3. Check is p prime or not, if yes GO TO 8 Step else GO TO 3;
4. Find p_i;
5. Compute k;
6. Find e_i;
7. Return with p, p_i, k, and e_i
8. Return with p, $p_i = p$, k = 1, and $e_i = 1$

3.1.2 Computing NPP

Input: n; Function: A.1; generates p, k, and e_i; Output: NPP

1. Use the Equations (1) and (2) to compute NPP
2. Return with NPP

3.1.3 Computing Irreducible Characteristic Polynomials

Input: n; Function: A.1; generates p, k, and e_i; Output: List of irreducible characteristic polynomials [irp]

1. Read n and do the followings
2. Write first in decimal, the digits 3 to p; and, then convert it into binary; delete all such entries which has total odd number of ones; and, call it CONE of size let mxn;
3. Read CONE; For i = 1:m construct matrix [A]; and check that for any factors of p, is $[A]^{p_i} = I$; if yes; GO TO next i; else the i^{th} vector of the CONE will be listed in [irp];
4. Compute the size of [irp], say jxn;
5. Return with [irp]

3.1.4. Computing Primitive Characteristic Polynomials [pp]

Input: n; Function: A.3; generates [irp]; Output: List of primitive characteristic polynomials [pp]

1. Read n and do the followings
2. Check for each [irp] entries [A] = $[A]^p$ = I, list this irp in the list of primitive polynomial [pp]; add it's reciprocal also in the list.
3. Return with [pp]

Example 3.1

The algorithm m is tested for different values of n, here for the purpose of demonstration, we present a case for n = 6. The computation debug is as below: Algorithm A1 computes as:

p = 63, p_i = {3, 7}, k = 2, and e_i = {2, 1}.

Algorithm 3.1.2 computes NP = 6.

Total possible polynomials NP = 32;

Algorithm 3.1.3 enumerates the following matrices.

CONE lists a total 15 elements as:

CONE = {000011, 000101, 001001, 001111, 010001, 010111, 011011, 011101, 100001, 100111, 101011, 101101, 110011, 110101, 111001}

And [irp] as,

[irp] = {000011, 000101, 001111, 010001, 011011, 011101, 100001, 100111, 101011, 101101, 110011, 111001}.

The debug results while implementing Algorithm 3.1.4 and after j = 7 the [pp] can be read as

[pp] = {000011, 100001, 011011, 101101, 100111, 101011}.

4 Simulation Result

Using the devised algorithm we run our program and due to the space problem, only a sample of some of the results is demonstrated below in Table 2.

Further, the devised program is capable of providing the sets of dense or sparse primitive characteristic polynomials, for example for n = 32, a set of these are provided below:

Table 2: Some of the outputs of the program

n	p	p_i	NPP
24	16777215	3^2, 5, 7, 13, 17, 241	276480
30	1073741823	3^2, 7, 11, 31, 151, 331	17820000
32	4294967295	3, 5, 17, 257, 65537	67108864
33	8589934591	7, 23, 89, 599479	211016256

Sparse with minimum feedback taps:

$x^{32}+x^7+x^6+x^2+1$; $x^{32}+x^8+x^5+x^2+1$; $x^{32}+x^9+x^3+x^2+1$;

Sparse with 6 feedback taps:

$x^{32}+x^7+x^5+x^3+x^2+x+1$; $x^{32}+x^7+x^6+x^5+x^4+x^2+1$;

Sparse with maximum feedback taps:

$x^{32}+x^{31}+x^{30}+x^{29}+x^{28}+x^{27}+x^{26}+x^{25}+x^{24}+x^{23}+x^{22}+x^{21}+x^{20}+x^{19}+x^{18}+x^{17}+x^{16}+x^{15}+x^{14}+x^{13}+x^{12}+x^{11}+x^{10}+x^9+x^8+x^7+x^6+x^5+x^4+x^2+1$;

$x^{32}+x^{31}+x^{30}+x^{29}+x^{28}+x^{27}+x^{26}+x^{25}+x^{24}+x^{23}+x^{22}+x^{21}+x^{20}+x^{19}+x^{18}+x^{17}+x^{16}+x^{15}+x^{14}+x^{13}+x^{12}+x^{11}+x^{10}+x^9+x^8+x^7+x^6+x^4+x^3+x^2+1$;

Sparse with 26 feedback taps:

$x^{32}+x^{31}+x^{30}+x^{29}+x^{28}+x^{27}+x^{26}+x^{25}+x^{24}+x^{23}+x^{22}+x^{21}+x^{20}+x^{19}+x^{18}+x^{17}+x^{16}+x^{15}+x^{14}+x^{13}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^2+1$;

$x^{32}+x^{31}+x^{30}+x^{29}+x^{28}+x^{27}+x^{26}+x^{25}+x^{24}+x^{23}+x^{22}+x^{21}+x^{20}+x^{19}+x^{18}+x^{17}+x^{16}+x^{15}+x^{14}+x^{13}+x^{12}+x^{11}+x^{10}+x^8+x^6+x^5+1$;

Sparse with 28 feedback taps:

$x^{32}+x^{31}+x^{30}+x^{29}+x^{28}+x^{27}+x^{26}+x^{25}+x^{24}+x^{23}+x^{22}+x^{21}+x^{20}+x^{19}+x^{18}+x^{17}+x^{16}+x^{15}+x^{14}+x^{13}+x^{12}+x^{11}+x^{10}+x^9+x^7+x^6+x^5+x^2+1$;

$x^{32}+x^{31}+x^{30}+x^{29}+x^{28}+x^{27}+x^{26}+x^{25}+x^{24}+x^{23}+x^{22}+x^{21}+x^{20}+x^{19}+x^{18}+x^{17}+x^{16}+x^{15}+x^{14}+x^{13}+x^{12}+x^{11}+x^{10}+x^9+x^5+x^4+x^3+x^2+1$;

4 Conclusion

Through this work we wanted to demonstrate an algorithmic procedure for computing the primitive polynomials for the use of CRC applications. The presented devised algorithmic procedure is the results based on the exhaustive research work on the theory of the primitive polynomials and the related mathematics. The design of the algorithm is such that it can provide the different kind of the subsets

of the primitive polynomials. These subsets can be obtained in groups by reserving the different kinds of parameters like the number of the taps, a particular tap, minimum number of taps, and maximum number of taps, security level, power consumption, time requirements, silicon area etc. Hence, the algorithm makes it possible to provide an optimal CRC designs to meet the specified requirements.

ACKNOWLEDGMENT

The acknowledgements are due to Sultan Qaboos University (Sultanate of Oman) for providing research support grant (SQU-DVC/ PSR/RAID/2010/23).

References:

[1] Ruckmani, S.R., and Anbalagan, P., 'High Speed cyclic Redundancy Check for USB', DSP Journal, vol. 6, no. 1, Sept., 2006, pp. 45 - 50

[2] Koopman, P., '32-bit cyclic redundancy codes for Internet applications', Proceedings. International Conference on Dependable Systems and Networks, 2002 (DSN 2002), vol., 2002, pp. 459 – 468

[3] J. A. Davis, M. Mowbray, and S. Crouch, 'Finding cyclic redundancy check polynomials for multilevel systems', IEEE Trans on communications, vol. 46, no. 10, Oct.1998, pp. 1250 -53

[4] http://en.wikipedia.org/wiki/Cyclic_redundancy_check

[5] Krogsgaardt, K. and Karp, T., 'Fast identification of primitive polynomials over Galois fields', Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'05), 18-23 March 2005, vol. 5, pp. 553 – 556.

[6] S. Maitra, K. C. Gupta, and A. Venkateswarlu, Results on multiples of primitive polynomials and their products over GF(2)', Theoretical Computer Science (Elsevier Science Publishers Ltd.), vol. 341 , no.1, Sept. 2005, pp. 311 – 343.

[7] Pradhan, D.K, Kagaris, D. and Gambhir, R., 'A Hamming distance based test pattern generator with improved fault coverage', 11th IEEE International Testing Symposium, vol., 6-8 July 2005, pp. 221 – 226

[8] Udar, S., Kagaris, D., 'LFSR Reseeding with Irreducible Polynomials', 13th IEEE International On-Line Testing Symposium, 2007 (IOLTS 07), vol., 8-11 July 2007, pp. 293 - 298

[9] Green, D.H., Amarasinghe, S.K., 'Sequences and arrays derived from non-primitive irreducible polynomials', IEE Proceedings Computers and Digital Techniques, vol. 139, no. 4, Jul 1992, pp. 363 – 371

[10] Ahmad A., Nanda N.K. and Garg K., 'Are primitive polynomials always best in signature analysis?', IEEE design & Test of Computers (USA), 1990, vol.7, no.4, pp. 36 - 38

[11] Ahmad A. and Elabdalla A. M., 'An efficient method to determine linear feedback connections in shift registers that generate maximal length pseudo-random up and down binary sequences," Computer & Electrical Engineering -An Int'l Journal (USA), 1997, vol. 23, no. 1, pp. 33-39.