http://www.zju.edu.cn/jzus; E-mail: jzus@zju.edu.cn
ISSN 1009-3095 *Journal of Zhejiang University SCIENCE* V.4, No.4, P.433-436, July-Aug., 2003

An efficient method for parallel CRC automatic generation*

CHEN Hong-sheng (陈红胜)†, ZHANG Wei-cheng (张维承),
WANG Yong (王勇), CHEN Kang-sheng (陈抗生)

(*Department of Information Science and Electronic Engineering, Zhejiang University, Hangzhou 310027, China*)

†E-mail: hansomchen@zju.edu.cn

Received July 6, 2002; revision accepted Nov.21, 2002

Abstract: The State Transition Equation (STE) based method to automatically generate the parallel CRC circuits for any generator polynomial or required amount of parallelism is presented. The parallel CRC circuit so generated is partially optimized before being fed to synthesis tools and works properly in our LAN transceiver. Compared with the cascading method, the proposed method gives better timing results and significantly reduces the synthesis time, in particular.

Key words: State Transition Equation (STE), CRC, Linear Feedback Shift Register (LFSR)
Document code: A **CLC number:** TN919.3; TN929

INTRODUCTION

CRC (Cyclic Redundancy Checking) codes are widely used in telecommunication, especially in the internal layers of protocols such as Ethernet, X25, FDDI and ATM (AAL5). The common hardware solution for CRC calculation is the linear feedback shift register (LFSR), characterized by a simple bit-serial architecture for both coding and decoding messages. It is difficult to improve data throughput properly by this architecture, which leads to parallel solutions for CRC circuit. Several parallel solutions for CRC calculation, including hardware solutions and software solutions based on table look up algorithms, have been proposed by many authors such as Pandeya *et al*.(1975), Perez (1983), Sarwate (1988), Ramabadran *et al*. (1988), Glaise *et al*. (1993), Matsushima *et al*. (1996), and so on. Pei *et al*.(1992) derived the parallel CRC circuit from the LFSR's state transition equation, assuming that the generator polynomial and the number of bits processed in parallel is given. According to our knowledge, automatically generating the CRC circuits has not been studied in detail so far.

Sprachmann (2001) presented a generic VHDL description to generate parallel CRC circuits automatically characterized by the basic LFSR's cascading. However, it is inconvenient for some synthesis tools to generate and optimize the final feedback network of the CRC circuit.

The automatic generation of parallel CRC circuits presented in this paper is featured by an STE-based method. The process is easy to understand. Compared with the method presented by Sprachmann, the CRC block generated by our approach has better timing results and, in particular, costs much less synthesis time. As a tradeoff, this approach needs higher area consumption.

PRINCIPLE

Using parallel CRC circuit can efficiently improve data throughput. A typical parallel CRC circuit is shown in Fig.1.

The contents of the state registers depend on their initial value and the subsequent message bits. This generates a large XOR network. How to automatically generate this network and provide VLSI designers a simplified XOR network is

* Project (No.60271010) supported by the National Natural Science Foundation of China

the focus point of this paper. Differing from Sprachmann, we aim to leave the synthesis tools a partially optimized XOR network, rather than a mass of non-optimized logic block. Obviously, this approach would save much synthesis time. It also hides heavy algebra from designers.

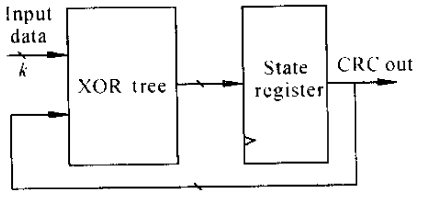

Fig.1 A typical parallel CRC architecture

We use STE of parallel CRC circuit to simplify the XOR network. Let $\boldsymbol{S}(t)=[s_0\ s_1\cdots s_{n-1}]$ be the remainder after t conditional subtractions and $\boldsymbol{G}=[g_0\ g_1\cdots g_{n-1}]$ be the coefficient vector of nth-order generating polynomial. For a given parallelism k, assume $\boldsymbol{Z}(t)=[Z_t\ Z_{t+1}\cdots Z_{t+k-1}]$ to be a vector containing a parallel group of k message bits and we have the STE of the parallel CRC circuit:

$$\boldsymbol{S}(t+k)=[s_0\ s_1\ldots s_{n-k-1}][\boldsymbol{0}\,|\,\boldsymbol{I}_{n-k}]\oplus([s_{n-k}\ s_{n-k-1}\ldots s_{n-1}]\oplus\boldsymbol{Z}(t))\boldsymbol{D} \quad (1)$$

where s_i is shorthand for $s_i(t)$, $\oplus$ represents modulo-2 addition,

$$[\boldsymbol{0}\,|\,\boldsymbol{I}_{n-k}]=\begin{bmatrix}0&\cdots&0&1&0&\cdots&0\\0&\cdots&0&0&1&\ddots&\vdots\\\vdots&\vdots&\vdots&\vdots&\ddots&\ddots&0\\0&\cdots&0&0&\cdots&0&1\end{bmatrix}$$

is a $(n-k)\times n$ *matrix and*

$$\boldsymbol{D}=\begin{bmatrix}\boldsymbol{G}\\\boldsymbol{GT}^1\\\cdots\\\boldsymbol{GT}^{k-1}\end{bmatrix} \quad (2)$$

where $\boldsymbol{T}^j$ is the jth power of the matrix

$$\boldsymbol{T}=\begin{bmatrix}0&1&0&.&0\\0&0&1&.&0\\.&.&.&.&1\\g_0&g_1&g_2&.&g_{n-1}\end{bmatrix} \quad (3)$$

From Eq.(1) we can generate a partially optimized XOR network for any parallelism and any generator polynomial. Before the XOR network is fed to the synthesis tools for optimization, its complexity is considerably reduced by applying the following simple rules:

$$a\ \text{XOR}\ a\ =\ 0\ \text{and}\ a\ \text{XOR}\ 0\ =\ a.$$

This task, which might be very time-consuming, is done by synthesis tools as proposed in Sprachmann's (2001).

REALIZATION

To be accepted by VLSI designers, the parallel generic approach for CRC circuits must,

1. hide heavy algebra from the designer,
2. be universally valid and therefore generic with respect to generator polynomials and degree of parallelism, and
3. be described in a standard hardware description language (VHDL or Verilog) to be used by VLSI synthesis tools.

Ying (2000) presented a survey on software evolution and computer-aided prototyping. Using STE-based method, we provide a program with a simple, user-friendly interface to realize parallel CRC circuits automatic generation. It is parameterizable for various polynomials and adjustable in the amount of parallelism. It also supports CRC reset and enable functions. The algebraic calculation in Eq.(1) is processed in the background. The designer need not be concerned with the internal implementation. According to the user's choice, the parallel CRC circuits can be described in either Verilog or VHDL to be conveniently interpreted by VLSI synthesis tools.

SYNTHESIS RESULTS

The architectures generated by the proposed method and cascading method have both been implemented on FPGA devices of the SPARTAN2 XILINX family. The generator polynomial is shown in Eq.(4), which is used on Ethernet, FDDI and AAL5-ATM. The synthesis was done using Synopsys FPGA Express 3.5 and XILINX Foundation series 3.1. The architecture tested in these examples implements a reset operational CRC checker. The results of 4 different levels of parallelism are given in Table 1, including synthesis time, maximum frequency (indicate tim-

ing results) and equivalent gate count.

It can be seen from Table 1 that cascading strategy costs less area consumption while STE-based strategy gives better timing results. We note that the ratio of the maximum frequency to area cost achieved by our method is smaller than that by the Sprachmann method when the parallelism k is small. We interpret this to be that optimizing is much difficult when the architecture is simple to synthesize because it almost reaches its upper speed limit. This can be verified by the condition when k is equal to 32. In this situation the architecture is more complicated. The performance achieved by our method is much better than that achieved by cascading method. Moreover, STE-based strategy also involves less synthesis time, especially as the parallelism goes up.

Table 1 Comparison of the synthesis cost

CRC Parallelism k (bit)	4	8	16	32
Synthesis time[a]/ Maximum Frequency / Area with STE-based approach presented by this paper	6s / 146.5MHz / 529egc[b]	11s / 134MHz / 790egc	24s / 130MHz / 1273egc	62s / 121.5MHz / 2212egc
Synthesis time[a]/ Maximum Frequency / Area with Cascading method presented by Sprachmann	10s / 155MHz / 484egc[b]	87s / 130MHz / 598egc	310s / 106MHz / 838egc	1630s / 66MHz / 1330egc

[a] Synthesis time on x86 family processor with 366-MHz CPU clock rate and 128M memory.

[b] Equivalent gate count, indicating the area consumption.

APPLICATION

This approach was successfully applied to an embedded internet system. In the embedded Internet system, we generated a CRC circuit to be used in a Local Area Network (LAN) transceiver. In the transmit function, for example, as seen in Fig.2 in our design, the parallel CRC circuit is generated as follows:

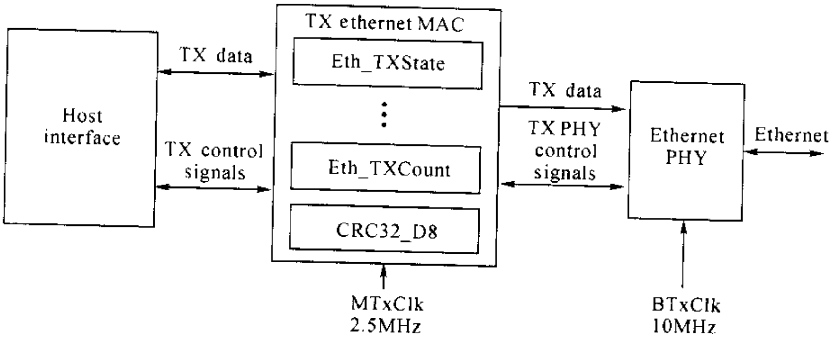

Fig.2 LAN transmit architecture

According to IEEE802.3 standard (2000 Edition), the generator polynomial is shown in Eq.(4). With the parallelism $k = 8$ and $n = 32$, we have Eq.(5) and the STE for this parallel CRC circuit becomes Eq.(6).

$$\boldsymbol{G} = [11101101101110001000001100100000] \tag{4}$$

$$\boldsymbol{D}=\begin{bmatrix}11101101101110001000001100100000\\01110110110111000100000110010000\\00111011011011100010000011001000\\00011101101101110001000001100100\\00001110110110111000100000110010\\00000111011011011100010000011001\\11101110000011100110000100101100\\01110111000001110011000010010110\end{bmatrix} \quad (5)$$

$$\boldsymbol{S}(t+8)=[00000000 s_0 \cdots s_{23}]\oplus [w_0 w_1 \cdots w_7]\boldsymbol{D} \quad (6)$$

where $w_i = s_{i+24} \oplus z_{t+7-i}$, for $0 \leqslant i \leqslant 7$.

We use Eq.(6) to generate the partially optimized parallel CRC circuit. This circuit works properly in our embedded Internet system, which was implemented in a FPGA chip. Parallel CRC encoding and decoding considerably improves the system performance.

CONCLUSIONS

We applied the state transition equation to the automatic generation of parallel CRC circuits. The designer can easily explore design alternatives by using it. The process hides algebraic transformations. The proposed method gives better timing results and significantly reduces the synthesis time compared with the cascading method, especially when the parallelism is high, since the CRC block automatically generated by our approach is partially optimized. The tradeoff is that the CRC circuit generated by our approach needs more chip area.

References

Glaise, R.J. and Jacquart, X., 1993. Fast CRC Calculation. IEEE International Conference on Computer Design: VLSI in Computers and Processors, p.602 – 605.

Matsushima, T.K., Matsushima, T. and Hirasawa, S., 1996. Parallel encoder and decoder architecture for cyclic codes. *IEICE Transactions fundamentals*, E79 – A(9):1313 – 1323.

Pandeya, A.K. and Cassa, T.J., 1975. Parallel CRC Lets many lines use one circuit. *Computer Design*, **14**(9): 87 – 91.

Perez, A., 1983. Byte-wise CRC calculation. *IEEE Micro*, **3**(3):40 – 50.

Pei, T.B. and Zukowski, C., 1992. High-speed parallel CRC circuits in VLSI. *IEEE Trans. Comm.*, **40**(4): 653 – 657.

Ramabadran, T.V. and Gaitonde, S.S., 1988. A tutorial on CRC computations. *IEEE Micro*, **8**(4):62 – 75.

Sarwate, D.V., 1988. Computation of cyclic redundancy Checks via table loop-up. *Communications of the ACM*, **31**(8): 1008 – 1013.

Sprachmann, M., 2001. Automatic generation of parallel CRC cricuits. *IEEE Design & Test of Computer*, **18**(3):108 – 114.

Ying, J., 2000. Research on computer-aided prototyping system and software evolution. *Journal of Zhejiang University SCIENCE*, **1**(4): 384 – 387.

IEEE Computer Society Technical committee on Computer Communications, 2000 Edition. IEEE Standards for Local Area Networks: Carrier Sense Multiple Access with Collision Detection (CSMA/CD): Media Access Control Frame Structure.