PARALLEL COMPUTATION OF CRC USING SPECIAL GENERATOR POLYNOMIALS

Hamed Sheidaeian[1] and Behrouz Zolfaghari[2]
[1] Department of Engineering, Islamic Azad University, Garmsar Branch, Iran
hsheidaeian@iau-garmsar.ac.ir
[2] Department of Computer Engineering, Amirkabir University, Tehran, Iran
zolfaghari@aut.ac.ir

ABSTRACT

CRC (Cyclic Redundancy Check) is an error detection method commonly used in data communication systems, computer networks and storage environments. In this method, the transmitter divides the message by an agreed upon polynomial called the generator and concatenates the calculated residue to the message. The properties of the generator determine the range of errors which are detectable in the receiver side. The division operation is currently performed using serial circuits called Linear Feedback Shift Registers especially in the Ethernet network access protocol. Developing methods for parallel computation of the residue makes CRC suitable for higher layer protocols and software applications. This paper studies a case for parallel CRC computation using special generators which have special multiples called OZO (One-Zero-One) polynomials are divisible. We first provide a systematic approach to finding such polynomials and then design and evaluate the algorithm and the hardware required to perform the parallel division.

KEYWORDS

Cyclic Redundancy Codes, parallel CRC, OZO polynomials

1. INTRODUCTION AND BASIC CONCEPTS

Before explaining the CRC and our proposed method, we need some basic concepts and some preliminary discussions which are presented in the following:

Polynomial notation: is a common notation used in the literature for representing bit strings. For example, the string 110111 is represented by the polynomial $P = x^5 + x^4 + x^2 + x + 1$. As seen in the example, polynomials of degree *n* are the representative for a string which is *n+ 1 bit* long. Thus polynomials of odd degrees (called odd polynomials) represent bit strings with even length (called even strings) and vice versa.

OZO Polynomials: are polynomials having a form like $P = x^n + 1$. Such polynomials represent bit strings like 100…001, hence the name OZO (One-Zero-One). If *n* is odd, the polynomial is called an odd OZO. Similarly, the polynomial is called to be even if *n* is even.

Modulo-2 Addition and Subtraction: are performed without generating carry or borrow bits. Modulo-2 addition and subtraction are both performed using logical XOR.
Modulo-2 multiplication: is performed through consequent shifts and modulo-2 additions.

Modulo-2 Division: is performed through consequently subtracting multiples of the divisor from the dividend. The modulo-2 subtractions are continued until the degree of the residue becomes smaller than that of the divisor. Figure 1 shows examples of modulo-2 addition/subtraction, multiplication and division.

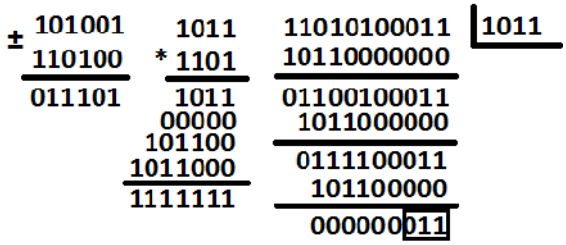

Figure 1: examples of modulo-2 addition/subtraction, multiplication and division

Concatenation: If S1 and S2 are two bit strings of degrees m and n, we will have $S1S2 = S1.2^n + S2$. Readers are referred to [7, 8] for more information regarding modulo-2 computations.

Cyclic Redundancy Code: The CRC works as follows. Whenever the sender has a message M to send, it first concatenates n zero bits to the right of the massage, converting it to $M.2^n$ (n is the length of an agreed-upon string called the *generator* subtracted by one. It is also the length of the CRC. Especially the Ethernet protocol uses a 32-bit CRC [22]). The sender divides the produced string $(M.2^n)$ by the generator (G) in the next step and calculates the residue $(R = (M.2^n) Mod\, G)$. Then the residue is replaced for the n zero bits. The string is now converted to $M' = MR = M.2^n + (M.2^n) Mod\, G$. The string $M' = MR$ is transmitted instead of M. Figure 2 shows these steps.

The addition, multiplication and division operations are performed modulo-2 here. Since addition and subtraction are the same in modulo-2 computations, we can think of M' as $M' = MR = M.2^n - (M.2^n) Mod\, G$ which is obviously divisible by G.

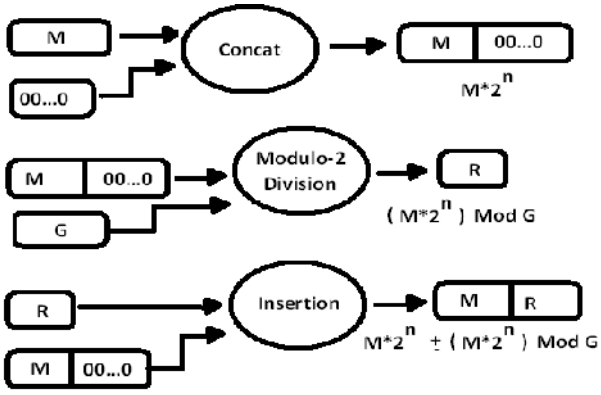

Figure 2: The transmitter side CRC Process

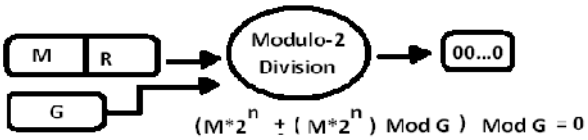

Figure 3: The Receiver side CRC Process

The receiver divides what it receives by G again and calculates the residue. If the receiver gets exactly the string transmitted by the sender, the residue will obviously be equal to zero. Figure 3 shows this process. But if an error has occurred through the channel, we can model the error as a string E added to M' [7, 8]. In such a case, the receiver receives $M' + E$ instead of M'. Since M' is divisible to G, the calculated residue in this case will be equal to $(M' + E)\ Mod\ G = E\ Mod\ G$. Applications of CRC [12, 17, 18, and 21] as well as developing methods for improving its efficiency [13, 19, 20, and 23] have been research focus in recent years. CRC is traditionally computed by serial circuits called LFSRs (Linear Feedback Shift Registers). An LFSR is a special kind of shift register in which the output of the last flip flop is fed to the input of the first flip flop through a number of XOR gates. Fig.4 shows a sample LFSR.

This paper proposes a novel method for parallel computation of CRC using mathematical properties of a special category of generator polynomials called ODPs (OZO Dividing Polynomials). ODPs are polynomials having multiples of form 100…001. The latter form of polynomials is called OZO (One-Zero-One). We demonstrate that if the generator is selected from this category, the CRC can be calculated by parallel circuits with minor hardware requirements. Zolfaghari, et al. [7, 8] introduced OZOs and ODPs. They developed a systematic method for constructing ODP polynomials. The rest of this paper is organized as follows. Section 2 presents some preliminary discussions, section 3 examines related works and section 4 presents the proposed method. Section 5 is dedicated to conclusions and further works.

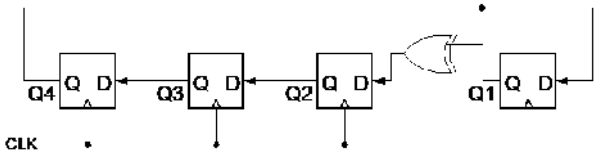

Figure 4: A sample LFSR

2. PRELIMINARY DISCUSSIONS

The main idea behind our proposed approach is performing a number of operations (each of which takes a cycle in the traditional modulo-2 division circuits) in a single cycle. We use the properties of OZO strings to achieve this goal in this paper. Let us clarify the approach by an example. Suppose that the divisor is *G=1111*. This string is an ODP. In fact, if we multiply this string by 11 we will get *G'=1111.11=10001*. It is easy to see that *G'* is an *m=5* bits long multiple of *G*. Thus every string can be divided by *G* through subtracting multiples of *G'* from the dividend. Figure 5 shows two steps such a division. The two steps shown in this figure convert 4 consequent bits from the left of the dividend to zero. These steps take 4 cycles by an LFSR. Our approach performs the operations of these two steps in a single cycle.

Let us see how the approach works. The dividend *D* is *n=10* bits long in this example. A close look shows that in the first step, the leftmost 1 of the divisor (G'_{m-1}) has been put under the leftmost 1 of the dividend $D_{n-1}(D_5)$.This causes the bit (D_{n-m}) to be inverted (XORed with 1).

Meanwhile, D_{n-1} has been converted from 1 to 0. This step has not affected $D_{n-m}, D_{n-m+1}, D_{n-2}$. The reason is that the corresponding bits of G' are equal to 0. Now let us look at the second step. In this step, G'_{m-1} has been put under $D_{n-3}$and D_{n-m-2} has been XORed with 1. In this step, D_{n-3} has been converted from 1 to 0. We can summarize the operations performed in the 2 steps (4 cycles) as follows. For each $D_i\ n-m < i < n-1$ if $D_i = 1$ then D_i is converted to 0 and D_{i-m+1} is XORed with 1 (XORed with D_i). In other words, if we divide *D* into *m-1* bit segments, the segment before the last has been XORed with the last segment and the last segment has been converted to 0. This process has reduced the length of *D* by m-1. Going on this procedure until the length of *D* is decreased less than that of *G* will lead to the residue remaining t the last step.

```
1 0 1 0 0 1 0 1 0 1 | 1 1 1 1
1 0 0 0 1
    1 0 0 0 1
-------------------
0 0 0 0 1 1 1 1 0 1
```

Figure 5: Two steps of division by an OZO divisor

3. RELATED WORKS

Improving the performance of CRC computations has been the topic for a lot of research in recent years [1-6]. But the most relevant category of related works are those which have proposed parallel algorithms for calculating CRC. A number of works in this category are introduced in below.

Nguyen [9] argued that CRC has notable error detection capabilities but its calculation is slow in contrast to other error detection methods and the reason is that it depends on sequential polynomial division operations. He attempted to find CRCs that can be calculated fast. He found such CRCs and also proposed a method for calculating them. In his method CRC can be calculated without the need for table lookup. His method can totally avoid polynomial divisions or reduce the number of them.

Youngju, et al. [11] proposed a software parallel method for calculating CRC and named it N-byte RCC (Repetition of Computation and Combination). Their method depends on dividing the message to 4-bit chunks and table lookup. This method can be implemented using any off-the-shelf processor. They tested their method on 1-star NOCs and single Bus architectures.

Kounavis, et al. [10, 15] proposed a frame fork for designing a family of parallel CRC algorithms. They showed that algorithms designed in this framework can reduce the memory required for calculating CRC and make it convenient for different computer system architectures. They claimed that these algorithms can be implemented in software using any general purpose processor and without the need for any extra special hardware.

Zhanqi, et al. [14] presented a parallel algorithm for CRC calculation based on mathematical equations which can be implemented for different generator polynomials. They claimed that their algorithms are quite convenient for hardware implementation. They implemented their algorithm in hardware and achieved the input data rate of 21 Gbs. They showed the efficiency of their algorithm for 10G Ethernet as well as ATM. Ji, et al. [16] presented a method based on Galois

Fields multiplication and accumulation operations for CRC calculation. This method can gain unlimited speedup over serial methods and lookup-driven methods but increase area due to the need for GFMAC (Galois Field Multiplication and Accumulation) modules. Their method can calculate CRC in two or three cycles for every message length. They implemented their method using a reconfigurable processor to which the instructions required to perform a number of simultaneous GFMAC operations can be added. A sample implementation of this method which uses 4 GMAC modules can calculate 32-bit CRC for a 16-bye message in two or three cycles.

4. THE PROPOSED APPROACH

According to the discussions presented in section 2, our proposed approach works as follows. Suppose we are going to divide an *n* bit dividend by an *m* bit divisor and calculate the residue. The dividend is first partitioned into *m-1* bit segments. Then in each step, the segment before the last is XORed with the last segment and the last segment is truncated from the dividend. The division is finished when only one segment remains from the dividend. Figure 6 shows the block diagram of the circuit which implements our approach for a 64 bit dividend and a 9 bit divisor.

As shown in Fig. 6-a, this circuit is constructed from two registers R and S. S contains the last 8 bit segment and R contains the remaining 58 bits. Both registers are built using D flip flops. The functions of R and S each include 2-cycle periods. In the first period, the leftmost segment of R is XORed with S. In this half cycle, S remains unchanged. This is called the XOR half cycle. In the second half cycle, S is loaded with the leftmost segment of R. In this step, each flip flop *i* of R is loaded with output of flip flop number *i-8*. This is called the shift half cycle.

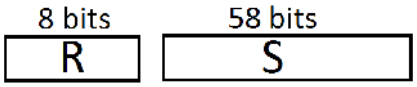

Figure 6.a: The shift registers

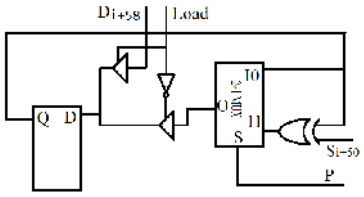

Figure 6.b: Feeding D flip flop number i from R

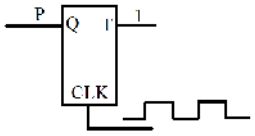

Figure 6.c: The circuit producing signal P

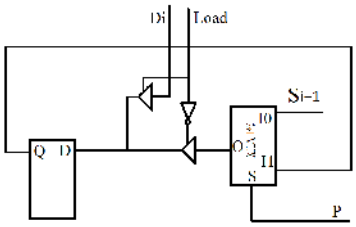

Figure 6.d: Feeding D flip flop number i from R

Fig. 6-b shows how D flip flop number I from R is fed. The signal P and the MUX allow the flip flop to function as needed for the division. During the XOR half cycle, each flip flop is loaded with the output of the XOR gate. During the shift half cycle, each flip flop is loaded with the output of another flip flop. Fig. 6-c shows the circuit producing the signal P which is built using a T flip flop. Fig. 6-d shows the feeding of flip flops of S. During the shift half cycle, these flip flops are loaded with corresponding flip flops of R. During the XOR half cycle, these flip flops remain unchanged.

The division takes $^{n}/_{m-1} - 1 = \frac{n-(m-1)}{m-1}$ cycles using the proposed approach while it takes *n* cycles using the traditional LFSR structure. In other words this circuit accomplishes the division $T = \frac{n}{\frac{n-(m-1)}{m-1}} = \frac{n(m-1)}{n-(m-1)}$ times faster than its equivalent LFSR. We refer to *T* as the PIR (Performance Improvement Ratio). Now let us examine the impacts on *m* and *n* on the PIR. Equation 1 gives the derivative of *T* to *n*.

$$^{d}/_{dn}\, T = \frac{-(m-1)^2}{(n-(m-1))^2} < 0 \qquad \textit{Equation 1}$$

This means that the PIR decreases with the increase of *n*. But the limit of *T* when n approaches infinity is equal to *m-1*. This means that the minimum of the PIR will be equal to *m-1*.
Equation 2 gives the derivative of *T* to *n*.

$$^{d}/_{dm}\, T = \frac{-n}{(n-(m-1))^2} < 0 \qquad \textit{Equation 2}$$

Again T is decreased with the increase of *m*. But here we should take into consideration that *m* cannot grow larger than *n*. We have coded this circuit with VHDL for a 24 bit dividend and 9 bit divisor. The division takes 2 cycles in this case. Fig. 7 shows the outputs of the circuit after the first and the second cycles.

5. CONCLUSION AND FURTHER WORKS

The CRC computation is traditionally implemented using sequential circuits called LFSRs. These circuits divide an n bit dividend by an m bit divisor in n cycles regardless of the size of the divisor. This paper showed that if the divisor is selected from a special family of strings called ODPs, the division can be implemented using a parallel circuit which takes $^{n}/_{m-1} - 1$ to accomplish the division. This circuit works $T = \frac{n(m-1)}{n-(m-1)}$ times faster than its corresponding LFSR. We designed the parallel circuit and coded with VHDL and reported the outputs of the circuit for a sample division. This work can be continued with designing parallel circuits for other families of divisors.

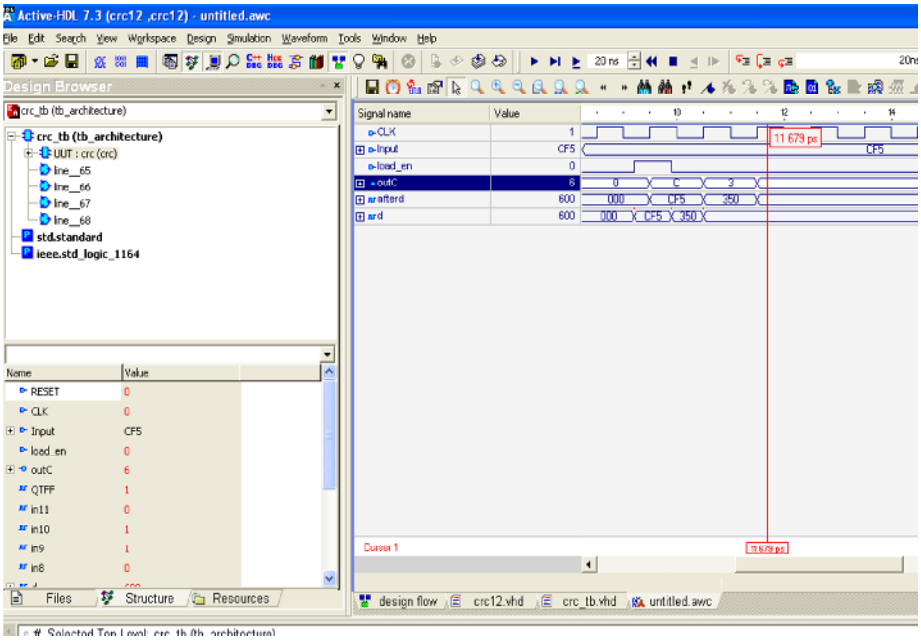

Figure 7-a: The output after the first cycle

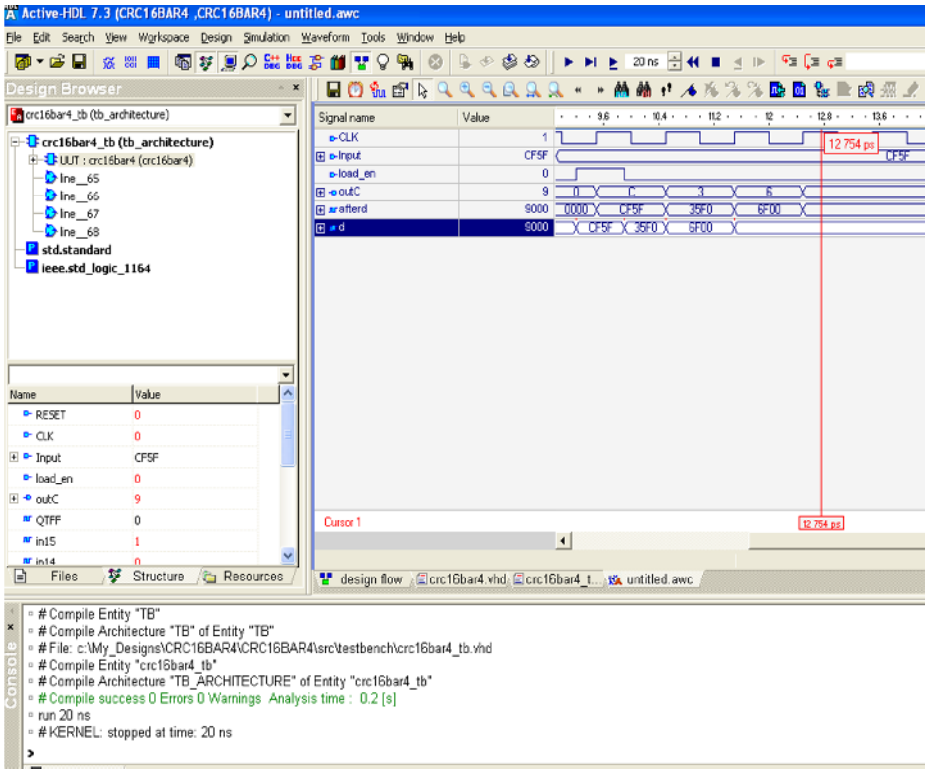

Figure 7-b: The output after the second cycle

REFERENCES

[1] Ustunel E., Hokelek I., Ileri O., Arslan H., Joint optimum message length and generator polynomial selection in cyclic redundancy check (CRC) coding, In Proceedings of 2011 IEEE 19th Conference on Signal Processing and Communications Applications (SIU), pp. 222 – 225, 2011

[2] Akagic A., Amano H., Performance evaluation of multiple lookup tables algorithms for generating CRC on an FPGA, In Proceedings of 2011 1st International Symposium on Access Spaces (ISAS), pp. 164 – 169, 2011

[3] Grymel M., Furber S.B., A Novel Programmable Parallel CRC Circuit, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Volume 19, Issue: 10, pp. 1898 – 1902, 2011

[4] Moon Jinyeong, Kih Joong Sik, Fast parallel CRC & DBI calculation for high-speed memories: GDDR5 and DDR4, In Proceedings of 2011 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 317 – 320, 2011

[5] Yanbin Zhang, Error correction application of CRC in the RFID system, In Proceedings of 2011 International Conference on Business Management and Electronic Information (BMEI),pp. 443-446, 2011

[6] Fouad, Marwa, Elsaddik, Abdulmotaleb, Using cyclic redundancy check to eliminate key storage for revocable iris templates, In Proceedin gs of 2011 24th Canadian Conference on Electrical and Computer Engineering (CCECE), pp. 117-120, 2011

[7] Behrouz Zolfaghari, Saadat Pour Mozaffari, Haleh Karkhane, A Systematic Approach to the Selection of CRC Generators to Detect Burst Errors in Ethernet Networks, IEEE International conference of Intelligent Network and Computing (ICINC 2010), Kuala Lumpur, Malaysia, November 2010

[8] Behrouz Zolfaghari, Hamed Sheidaeian, Saadat Pour Mozaffari, Systematic Selection of CRC Generator Polynomials to Detect Double Bit Errors in Ethernet Networks, 3rd International Conference on Computer Networks & Communications, Ankara, Turkey, 2011

[9] G.D. Nguyen, Fast CRCs, IEEE Transactions on, Vo. 58, No. 10, pp. 1321 – 1331, Oct. 2009

[10] Kounavis, Michael E. Berry, Frank L, Novel Table Lookup-Based Algorithms for High-Performance CRC Generation, IEEE Transactions on Computers, Vol.57, No 11, Nov. 2008

[11] Youngju Do, Sung-Rok Yoon, Taekyu Kim, Kwang Eui Pyun, Sin-Chong Park, High-Speed Parallel Architecture for Software-Based CRC, In Proceedings of International Conference on Consumer Communications and Networking (CCNC 2008), Las Vegas, NV, 10-12 Jan. 2008

[12] Iaodong Deng, Mengtian Rong, Tao Liu, Yong Yuan, Dan Yu, Segmented Cyclic Redundancy Check: A Data Protection Scheme for Fast Reading RFID Tag's Memory, In Proceedings of IEEE Wireless Communications & Networking Conference (WCNC 2008), pp. 1576-1581, March 31 2008 - April 3 2008, Las Vegas, Nevada, USA

[13] Walma Mathys, Pipelined Cyclic Redundancy Check (CRC) Calculation, In Proceedings of International Conference on Computer Communications and Networks, 2007 ICCCN 2007, In Proceedings of 16th International Conference on, pp 365-370, 13-16 August 2007. I. S.

[14] Xu Zhanqi, Yi Kechu, and Liu Zengji, A universal algorithm for parallel CRC computation and its implementation, Journal of Electronics (China), Vol. 23, No. 4, July, 2006

[15] Kounavis M.E., Berry F.L., A systematic approach to building high performance software-based CRC generators, In Proceedings of 10th IEEE Symposium on Computers and Communications (ISCC 2005), pp. 855-862, 2005

[16] Ji, H.M. Killian, E., Fast parallel CRC algorithm and implementation on a configurable processor, In Proceedings of 2002 IEEE International Conference on Communications (ICC 2002), pp. 1813-1817, 2002

[17] Jacobs and C. P. Bean, Fine particles, thin films and exchange anisotropy, in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350

[18] Ahmad, A. and Hayat, L., Algorithmic Polynomial Selection Procedure for Cyclic Redundancy Check for the use of High Speed Embedded Networking Devices, In Proceedings of International Conference on Computer and Communication Engineering 2008 (ICCCE'08), Kuala Lumpur, Malaysia - on 13-15 May, 2008

[19] Yun Pana, Ning Ge, Zaiwang Dong, CRC Look-up Table Optimization for Single-Bit Error Correction, Tsinghua University Journal of Science & Technology, Tsinghua Science & Technology, Vol. 12, Issue 5, pp. 620-623, October 2007

[20] Raman Assaf, Tyszberowicz Shmuel, The EasyCRC Tool, In Proceedings of 2007 International Conference on Software Engineering Advances (ICSEA 2007), pp. 25-31, August 2007
[21] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989
[22] Liu Zhanli, Liang Xiao, Zhao Chunming, Wang Jing, CRC-Aided Turbo Equalization For MIMO Frequency Selective Fading Channels, Journal of Electronics(China), Vol. 24, Issue 1, pp. 69-74, 2007
[23]Andrew. S. Tanenbaum, David J. Wetherall, Computer Networks, Prentice Hall,5th Edition, 2010

Authors

Hamed Sheidaeian has M.S. degree in computer engineering from Sharif University of Technology and is a faculty member of engineering department in Islamic Azad University, Garmsar Branch. His research areas include multimedia systems, computer graphic visualization, data communication and computer networks.

Behrouz Zolfaghari is a Ph.D. student in computer engineering at Amirkabir University of Technology (AUT), Tehran, Iran. He has published six journal papers and 16 conference papers by far. His research areas include image processing, computer architecture and computer networks.