Parallel Scrambler for High-Speed Applications

Chih-Hsien Lin, Chih-Ning Chen, You-Jiun Wang, Ju-Yuan Hsiao, and Shyh-Jye Jou

***Abstract*—In order to improve the speed limitation of serial scrambler, we propose a new parallel scrambler architecture and circuit to overcome the limitation of serial scrambler. A very systematic parallel scrambler design methodology is first proposed. The critical path delay is only one D-register and one XOR gate of two inputs. Thus, it is superior to other proposed circuits in high-speed applications. A new DET D-register with embedded XOR operation is used as a basic circuit block of the parallel scrambler. Measurement results show the proposed parallel scrambler can operate in 40 Gbps with 16 outputs in TSMC 0.18-μm CMOS process.**

***Index Terms*—Parallel scrambler, register, XOR.**

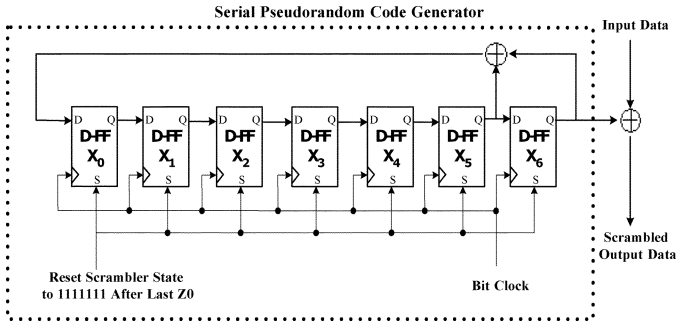

Fig. 1. Circuit diagram of serial scrambler in IEEE802.3ae.

I. Introduction

IN DIGITAL transmission systems, there are always scramblers to scramble the transmission data. In general, multiples of base rate signals are multiplexed and then scrambled before transmission which is descrambled and demultiplexed after reception. Scrambling used to be done serially. Standard like IEEE802.3ae (10-Gbps Ethernet) describes the functional diagram as a 7-bit series synchronous frame scrambler. The generated pattern is a maximal length sequence (m-sequence) of $2^N - 1$ (in this case $N = 7$). The scrambler diagram shown in Fig. 1 shall generate a continuous stream of output bits at the same rate as the transmitted bit rate (fs).

Nevertheless, as the operating frequencies of transmission systems grow beyond gigabits per second, serial scrambling techniques were no longer applicable. For example, in 10-Gbps Ethernet or 40-Gbps fiber transmission, with serial scrambling, this would mean working at frequency of 10/40 GHz which is not feasible with today's silicon-based CMOS integrated circuits. The requirement of high working frequency can be resolved by using parallel scrambling techniques [1]–[5] to enable the scrambling process at the low-frequency base rate. Under parallel scrambling, a set of scrambling processes are performed at the base rate, which collectively achieves the effect of serial scrambling when the scrambled base-rate signals are multiplexed to form a transmission-rate signal. A common characteristic of all well-known parallel solutions is that the number of inputs of the modulo-2 adders (XOR gates) used in the feedback loops of the pseudorandom code generator is more than two for some parallel ports. However, having to process the modulo-2 additions of more than two inputs will lead to an increase in the processing delay and lower the maximum working rate. Moreover, in today's deep submicrometer CMOS process, number of fanouts and interconnect length also become a significant factor that affect the processing delay. Thus, the architecture shall be regular and have less fanouts in the critical path.

In this brief, a very systematic parallel scrambler design methodology and new architecture is proposed. In the following, Section II will show the design methodology and procedures to develop parallel pseudorandom code generator used in parallel scrambling. Section III will show the architectures, circuits and measurement results. Finally, a conclusion is made.

Manuscript received March 31, 2004; revised December 1, 2005. This work was supported by the Chip Implementation Center, National Science Council and MOEC of Taiwan, R.O.C. under Grant NSC92-2215-E-008-003 and Grant 92-EC-17-A-07-S1-0001 This paper was recommended by Associate Editor M. Soma.

C.-H. Lin, C.-N. Chen, Y.-J. Wang, and J.-Y. Hsiao are with the Department of Electrical Engineering, National Central University, Jung-Li City, 320 Taiwan, R.O.C.

S.-J. Jou is Professor in the Department of Electronics Engineering, National Chiao Tung University, Hsinchu City, 300 Taiwan, R.O.C. (jerryjou@mail.nctu.edu.tw).

Digital Object Identifier 10.1109/TCSII.2006.875316

II. Realization of Parallel Pseudorandom Code Generator

There are various publications [1]–[5] in which parallel scrambling techniques are described. They allow any number of parallel bits to be generated in each clock cycle. To realize the parallel pseudorandom code generator, the minimized circuit complexity required N D-registers and M XOR2s (two inputs) [2], [5] for serial scrambler with a single XOR2. However, the XOR gates in the critical path for some of the parallel ports have more than two inputs. In the following, we will show the procedures that transform the serial scrambling to parallel scrambling with only a XOR2 gate in the critical path of the parallel ports.

(i) Describe the generating polynomial $P(x)$ as

$$P(x) = \sum_{q=0}^{N} c_q x^q \quad \text{or} \quad b(i) = \sum_{q=1}^{N} c_q b(i-q) \tag{1}$$

where $b(i)$ is the generated bits in the time sequence i of serial pseudorandom code generator with $c_N = c_0 = 1$

and $c_q = 0$ or 1 for other indexes. We can also write $b(\cdot)$ as

$$b(kN+i) = \sum_{q=1}^{N} c_q b((k-1)N + i + N - q) \quad \text{for } i = 0 \text{ to } N-1 \text{ and } k = 1 \text{ to } \infty. \quad (2)$$

Here, we define T as the minimal index of q that $c_q = 1$ and $D = N - T$. Also, S represents the number of coefficients that c_1 to c_N equal 1 (the number of inputs to the XOR gates). In applications, S is usually two to reduce circuit complexity.

Example 1: $P(x) = X^7 + X^6 + 1$ or $b(i) = b(i-7) + b(i-6)$ where $D = 7 - 6 = 1$ and $S = 2$. Substitute $N = 7$ into (2), we have $b(7k+i) = b(7(k-1)+i) + b(7(k-1)+i+1)$. □

(ii) Determine the number of parallel output port and write the parallel output bits generated in jth output cycle as a word B_j for $j = 0$ to ∞

$$B_j = [b_{Mj}, b_{Mj+1}, \ldots, b_{Mj+M-2}, b_{Mj+M-1}] \quad (3)$$

where b_{Mj+p} is the bit that generated in cycle j at port p. The initial conditions for $j = 0$ are stored in the registers when start-up.

Example 2:

$$\begin{aligned} B_0 &= [b_0, b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8, \\ &\quad b_9, b_{10}, b_{11}, b_{12}, b_{13}, b_{14}, b_{15}] \\ &= [x_6, x_5, x_4, x_3, x_2, x_1, x_0, x_6 + x_5, x_5 + x_4, \\ &\quad x_4 + x_3, x_3 + x_2, x_2 + x_1, x_1 + x_0, \\ &\quad x_6 + x_5 + x_0, x_6 + x_4, x_5 + x_3] \end{aligned}$$

where x_6 to x_0 represent the initial conditions of the serial pseudorandom code generator. □

Note that if $M < N$, we still need N registers to store the initial conditions.

(iii) For each bit b_{Mj+p} in B_j, calculate k and i by using $k = \lfloor (Mj+p)/N \rfloor$ and $i = (Mj+p)$ modulo N. Recursively apply (3) such that b_{Mj+p} only uses S bits in B_{j-1}.

Example 3:

$$\begin{aligned} B_1 &= [b_{16}, b_{17}, b_{18}, b_{19}, b_{20}, b_{21}, b_{22}, b_{23}, b_{24}, \\ &\quad b_{25}, b_{26}, b_{27}, b_{28}, b_{29}, b_{30}, b_{31}] \\ B_1^1 &= [b_9 + b_{10}, b_{10} + b_{11}, b_{11} + b_{12}, b_{12} + b_{13}, \\ &\quad b_{13} + b_{14}, b_{14} + b_{15}, b_{15} + b_{16}, b_{16} + b_{17}, \\ &\quad b_{17} + b_{18}, b_{18} + b_{19}, b_{19} + b_{20}, b_{20} + b_{21}, \\ &\quad b_{21} + b_{22}, b_{22} + b_{23}, b_{23} + b_{24}, b_{24} + b_{25}] \\ B_1^2 &= [b_9 + b_{10}, b_{10} + b_{11}, b_{11} + b_{12}, b_{12} + b_{13}, \\ &\quad b_{13} + b_{14}, b_{14} + b_{15}, b_8 + b_{10}, b_9 + b_{11}, \\ &\quad b_{10} + b_{12}, b_{11} + b_{13}, b_{12} + b_{14}, b_{13} + b_{15}, \\ &\quad b_{14} + b_{16}, b_{15} + b_{17}, b_{16} + b_{18}, b_{17} + b_{19}] \\ B_1^{2'} &= [b_2 + b_4, b_3 + b_5, b_4 + b_6, b_5 + b_7, b_6 + b_8, \\ &\quad b_7 + b_9, b_8 + b_{10}, b_9 + b_{11}, b_{10} + b_{12}, \\ &\quad b_{11} + b_{13}, b_{12} + b_{14}, b_{13} + b_{15}, b_{14} + b_{16}, \\ &\quad b_{15} + b_{17}, b_{16} + b_{18}, b_{17} + b_{19}] \\ B_1^4 &= [b_9 + b_{10}, b_{10} + b_{11}, b_{11} + b_{12}, b_{12} + b_{13}, \\ &\quad b_{13} + b_{14}, b_{14} + b_{15}, b_8 + b_{10}, b_9 + b_{11}, \\ &\quad b_{10} + b_{12}, b_{11} + b_{13}, b_{12} + b_{14}, b_{13} + b_{15}, \\ &\quad b_0 + b_4, b_1 + b_5, b_2 + b_6, b_3 + b_7] \\ B_1^{4'} &= [b_2 + b_4, b_3 + b_5, b_4 + b_6, b_5 + b_7, b_6 + b_8, \\ &\quad b_7 + b_9, b_8 + b_{10}, b_9 + b_{11}, b_{10} + b_{12}, \\ &\quad b_{11} + b_{13}, b_{12} + b_{14}, b_{13} + b_{15}, b_0 + b_4, \\ &\quad b_1 + b_5, b_2 + b_6, b_3 + b_7] \end{aligned}$$

where B_1^i is the word after applying (3) i times ($R = i$) for some bits in B_1. In this case, the $B_1^{4'}$ and B_1^4 are two solutions that only use 2 bits in B_0. □

(iv) Using the bit operations in B_1 as the logic operation in output ports.

Example 4: We can derive the output ports and their operations as

$$[p_0, p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}, p_{11}, p_{12}, p_{13}, p_{14}, p_{15}] = B_1^{4'} \text{ or } B_1^4.$$

□

This parallel architecture has the following properties.

1) Any number of parallel outputs can be designed from the serial scrambler. Recursive equations are derived to do the transformation.
2) If the number of inputs to the XOR of the serial scrambler is S (two in Fig. 1) then the number of inputs to the XOR in the parallel scrambler can also be S.
3) By applying recursion (3) different number of times, there are several representations that have the above two properties.

The relationships among N, D, M, and number of iterations (R) of applying (3) are shown in Fig. 2 and are written in the following propositions.

Proposition A: A parallel pseudorandom code generator with M parallel outputs has three cases for the number of register used according to the relationships of $M_{\max} = (N-D) \cdot R$, $M_{\min} = ((N+D) \cdot R)/2$ and R. i) One register at each port and total M registers. ii) More than one register at some ports and total N registers. iii) More than one D-register at some ports and total W registers.

W in case iii) of Proposition A depends on the two situations: 1) $W = R' \cdot N$ if $2M > R' \cdot N$ and (2) $W = [R' \cdot N - (R' \cdot N \bmod M)]$ if $2M \leqq R' \cdot N$. In here $R' = 2^{\lceil \log_2 (M/(N-D)) \rceil}$.

Proposition B: A parallel pseudorandom code generator with R equals to power of 2 has the property that each port only consists of $S - 1$ XOR2 gates.

Example 5: For polynomial $P(x) = X^7 + X^6 + 1$, $M_{\max}$, $M_{\min}$ and R are listed in Table I for R is power of 2. #

Example 6: $P(x) = X^7 + X^4 + 1$ is another polynomial of maximal length sequence of seven stage. In this case $N = 7$,

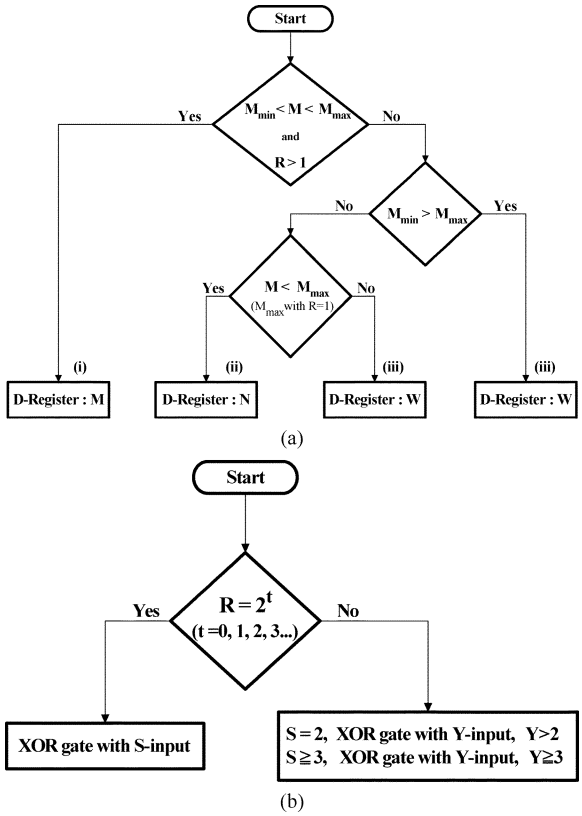

Fig. 2. (a) Total registers required. (b) Total XOR2 gates required.

TABLE I
R, M_{max} AND M_{min} FOR POLYNOMIAL $P(x) = X^7 + X^6 + 1$

R	2	4	8
M_{max}	12	24	48
M_{min}	8	16	32

$D = 3(D > N/3)$. For $M = 5$, we have

$$\begin{aligned}
B_0 &= [b_0, b_1, b_2, b_3, b_4] \\
&= [x_6, x_5, x_4, x_3, x_2] \\
B_1 &= [b_5, b_6, b_7, b_8, b_9] \\
B_1^1 &= [x_1, x_0, b_0 + b_3, b_1 + b_4, b_2 + b_5] \\
B_2 &= [b_{10}, b_{11}, b_{12}, b_{13}, b_{14}] \\
B_2^1 &= [b_3 + b_6, b_4 + b_7, b_5 + b_8, b_6 + b_9, b_7 + b_{10}] \\
B_2^2 &= [b_3 + b_6, b_4 + b_7, b_5 + b_8, b_6 + b_9, b_0 + b_6].
\end{aligned}$$

In B_1, b_9 generated at cycle 1 uses b_5 in the same cycle, thus it can not be implemented with only one XOR2 gate and one register. Thus, bits in B_1 are required to be stored as initial conditions

$$\begin{aligned}
B_0' = B_0 @ B_1 = \; & [x_6 @ x_1, x_5 @ x_0, x_4 @ (x_6 + x_3), \\
& x_3 @ (x_5 + x_2), x_2 @ (x_4 + x_1)] \\
[p_0, p_1, p_2, p_3, p_4] = B_2^2 &
\end{aligned}$$

where @ means cascade. With $M = 5$, it consists of five XOR2 gates with ten registers for storing initial conditions. However, the critical path is still one XOR2 gate and one register. □

Fig. 3(a) shows a parallel pseudorandom code generator with M equals 16 of the serial scrambling shown in Fig. 1 using B_1^4 in Example 3. Each output port consists of one XOR2 gate cascaded by one register and the number of fan-outs is from 2 to 5. This kind of parallel scrambler architecture is very regular and has the potential for high-speed operation. If maximal number of fanouts is very important, we can try to use $B_1^{4'}$ in Example 3 to reduce the maximal number of fanouts as shown in Fig. 3(b). As you can see, the maximal number of fanouts is reduced from 5 to 3. Fig. 4 shows another example of $P(x) = X^{11} + X^9 + 1$ used in IEEE1394b. Each output port only consists of one XOR2 gate cascaded by one register and the maximum number of fan-outs is 5. Fig. 5 shows the example used in Example 6 with five parallel outputs. Although each port required two registers, the critical path is still one XOR2 gate and one register. Fig. 6(a) shows the design example proposed in [5, Fig. 4] and Fig. 6(b) shows the one derived here. The comparisons are listed in Table II. Due to the reduction of two XOR2 to one XOR2, the proposed architecture can be used in higher operational speed.

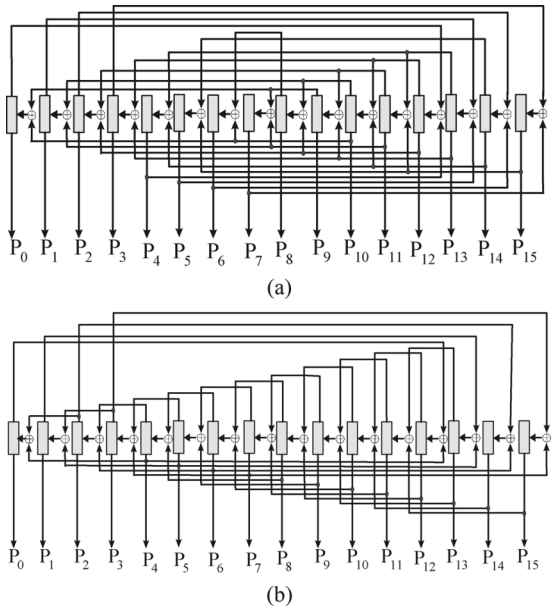

Fig. 3. Parallel pseudorandom code generators with $M = 16$. (a) Maximal fanouts of 5. (b) Maximal fanouts of 3.

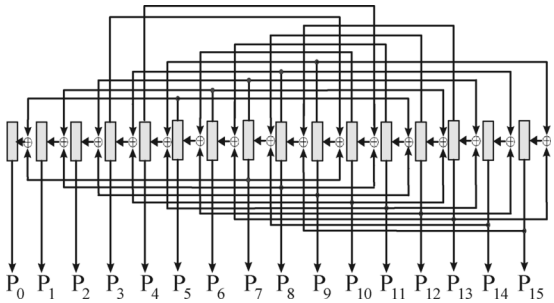

Fig. 4. Parallel pseudorandom code generators with $M = 16$ of $P(x) = X11 + X9 + 1$ used in IEEE1394.

III. Architecture and Circuit Design

In the parallel pseudorandom code generator, the basic circuit block is XOR2 gate cascaded by D-register. If single edge triggered (SET) D-registers are used, the operational frequency is fs/M. As we know, high-speed clock buffers consume lots of power due to the stringent timing requirement of rise/fall

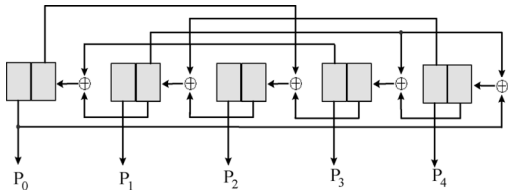

Fig. 5. Example used in Example 6 with 5 parallel outputs.

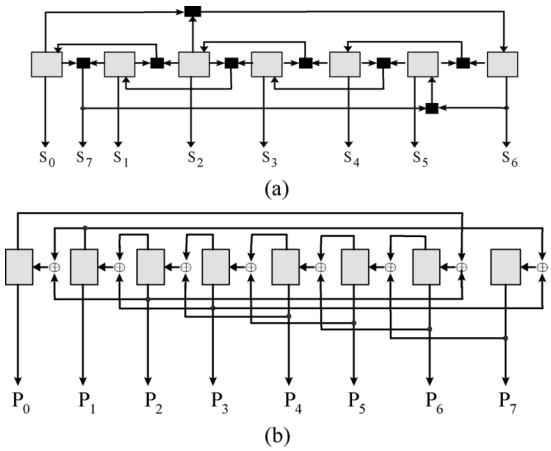

Fig. 6. $P(x) = X^7 + X^6 + 1$ with $M = 8$. (a) Circuit proposed in [5, Fig. 4]. (b) Circuit proposed in this brief.

TABLE II
COMPARISONS OF TWO DESIGNS OF $P(x) = x^7 + x + 1$ AND $M = 8$

	No. of registers	No. of XOR	Critical path	Fanouts
Fig.5(a)	7	8	one D-register and two XOR2 gate	4 (Max.)
Fig.5(b)	8	8	one D-register and one XOR2 gate	4 (Max.)

and delay time. Thus, we use double edge triggered (DET) D-register which provides data sampling at both rising and falling edges of clock. In this way, the clock frequency is only half of that used for SET D-register. Fig. 7(a) and (b) shows two conventional DET D-registers [6]. DET-TSPC merges two SET-TSPC D-registers but improves the design by reducing the number of clocked transistor to six instead of eight. It suffers from the same problems of TSPC-based register such as output dip, charge sharing, etc. DET-C^2MOS is a safety design and has low input loading (one pMOS and nMOS) as compared to DET-TSPC (two pMOS and two nMOS). Thus, it is ideally suited for two phase clock systems. The XOR circuit shown in Fig. 8(a) is a frequently used CMOS circuit. The critical path delay is only a pMOS or nMOS. The output is driven by signal source and is not driven by VDD and GND. This is one disadvantage of this XOR circuits. It will cause delay problem if such XOR's are cascaded. However, in here, no XORs are cascaded so it shall be no problem.

Table III shows the HSPICE simulation results of serial scrambler using the DET-TSPC or DET-C^2MOS cascaded by XOR2. The technology used is TSMC 0.18-μm 1.8-V CMOS process. The operational clock cycle $(T_{\rm clk})$ is limited by

$$T_{\rm clk/2} \geq T_{\rm clk-Q} + T_{\rm XOR} + T_{\rm setup} \quad (4)$$

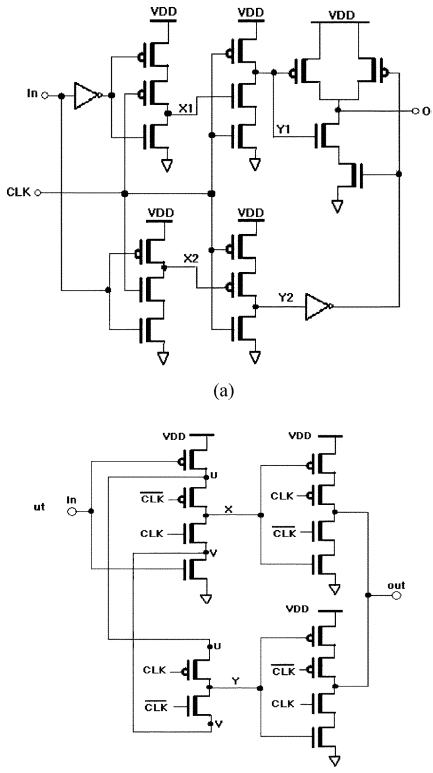

Fig. 7. (a) DET-TSPC. (b) DET-C^2MOS.

TABLE III
PERFORMANCE COMPARISONS OF SERIAL SCRAMBLERS

	DET-TSPC	DET-C^2MOS	XOR-DET-C^2MOS
Speed (Gbps)	2.50	2.56	3.60
Power (mW)	11.20	19.00	22.30
Power/speed	4.48	7.42	6.19

where $T_{\rm setup}$, $T_{\rm clk-Q}$, and $T_{\rm XOR}$ is the set up time, clock to output delay time of D-register, and delay time of XOR gate. We can reduce $T_{\rm XOR} + T_{\rm setup}$ by embedding the XOR operation into the master stage of DET-C^2MOS XOR-DET-C^2MOS as shown in Fig. 8(b). By doing so, not only the number of transistors is reduced by two but the delay path of XOR and set up path of DET-C^2MOS are merged. The XOR operation can not be merged into the DET-TSPC in the same way and need a much complicated structure. Table III also shows the simulation results of the serial scrambler using the proposed circuits. It can work up to 3.6 Gbps (1.8 GHz) and is 1.4 times faster than the conventional XOR cascaded by DET-C^2MOS. By using the proposed XOR-DET-C^2MOS circuit in the parallel scrambler [Fig. 3(a)], because the maximum fan-out number is 5 instead of 2 in serial scrambler, pre-layout simulations show that the operational clock can only work at 2.7 Gbps per port.

If a one-dimensional array like the one shown in Fig. 3(a) is implemented, the post-layout simulation shows that it can only work up to 2.4 Gbps. The decreasing of operational speed is due to a long interconnect of several hundreds of micrometers. We carefully redo the layout in a rectangular format (scrambler I) as shown in Fig. 9 to reduce the interconnect length. Post-layout simulation shows that it can work at 2.55-Gbps per port. Parallel

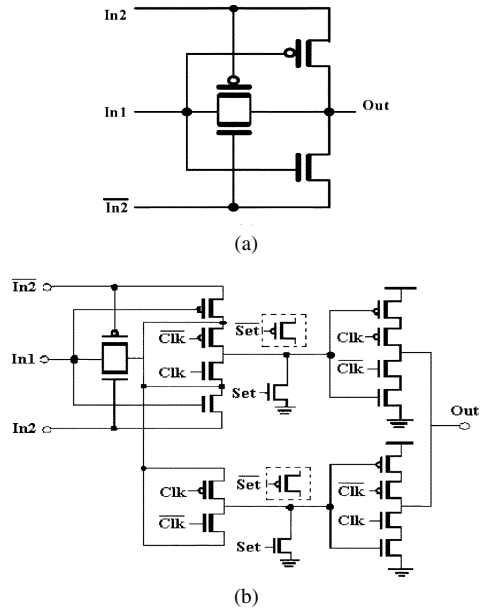

Fig. 8. (a) XOR circuit. (b) Proposed XOR-DET D-register.

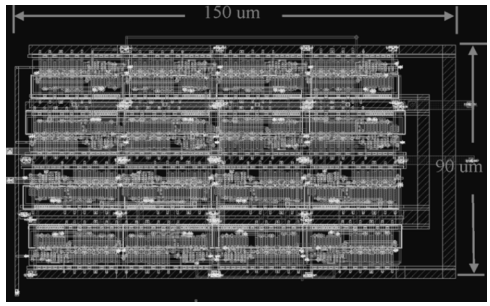

Fig. 9. Layout of the proposed parallel pseudorandom code generator.

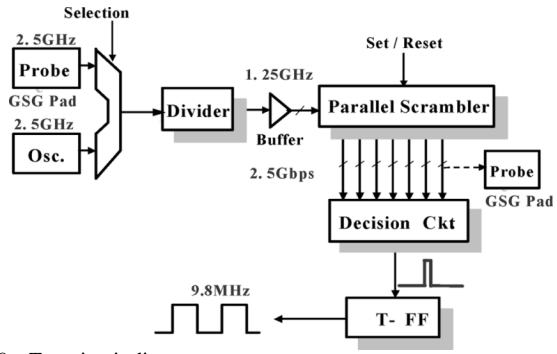

Fig. 10. Test circuit diagram.

scramblers designed by using XOR cascaded by DET-C^2MOS [Fig. 7(b), scrambler II] are also designed and implemented in a rectangular format. The post-layout simulation results show that the scrambler II can only work up to 1.76-Gbps per port. A test chip is implemented by using TSMC 0.18-μm 1.8-V CMOS process for both scrambler I and scrambler II as shown in Fig. 10. A divide-by-2 divider is used to have clock signal with 50% duty cycle. Due to the characteristic of the parallel scrambler, it has a periodic pattern every 127 output cycles. So we design a decision circuit that will be triggered each time this pattern is matched. Thus, when the output of the T-FF has a stable clock frequency of fs, then we know the parallel scrambler can work in fs*254*16 bps. The

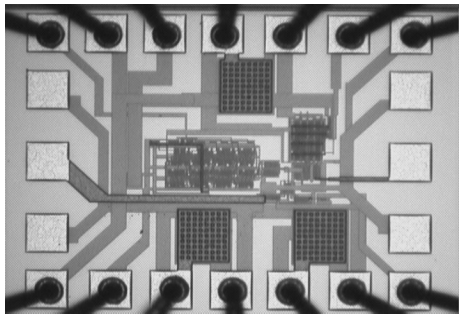

Fig. 11. Chip photo.

TABLE IV
CHIP SUMMARY AND COMPARISONS

	Post-Simulation		Measured Results	
	Scrambler 1	Scrambler 2	Scrambler 1	Scrambler 2
Data Rate (Gbps)	2.55	1.76	2.50	1.72
Area(um^2)	150 x 90	285 x 125	150 x 90	285 x 125
Power (mW)	27.61@ 27Gbps	32.47 @ 27Gbps	29.04@ 27Gbps	33.91@ 27Gbps
	34.12@ 40Gbps		40.49@ 40Gbps	

chip photo is shown in Fig. 11. The measurement results are listed in Table IV and quite match the post-layout simulation results. The measured maximum T-FF output frequency is 9.8448 MHz. Thus, the proposed parallel scrambler and new circuits can wok at 40 Gbps. The results show that the proposed circuits can work 1.5 times faster than the conventional one with smaller layout area and less power consumption.

IV. CONCLUSION

A very systematic parallel scrambler design methodology is proposed. The structure of the parallel scrambler is very regular and the critical path delay is only one D-register and one XOR2 gate. Moreover, by applying the recursive equations different number of times on different parallel output ports, we can have several representations of parallel scrambler with different number of fanouts and interconnect length. A new XOR-DET-C^2MOS cell is proposed to speed up the operation speed of the circuits. Measurement results show that the circuit is superior in speed than other designs. Design example shows that 40 Gbps with 16 outputs can be achieved by using 0.18-μm CMOS process.

REFERENCES

[1] D. W. Choi, "Parallel scrambling techniques for digital multiplexers," *AT&T Tech. J.*, vol. 65, pp. 123–136, Sept./Oct. 1986.
[2] W. J. McFarland, K. H. Springer, and C. S. Yen, "1-Gword/s pseudo-random woed generator," *IEEE J. Solid-State Circuits*, vol. 24, no. 3, pp. 747–751, Jun. 1989.
[3] S. W. Seetharam, G. J. Minden, and J. B. Evans, "A parallel SONET scrambler/descrambler architecture," in *Proc. IEEE ISCAS'93*, May 1993, vol. 3, pp. 2011–2014.
[4] B. G. Lee and S. C. Kim, "Low-rate parallel scrambling techniques for today's lightwave transmission," *IEEE Commun. Mag.*, pp. 84–95, Apr. 1995.
[5] S. C. Kim and B. G. Lee, "Realizations of parallel and multibit-parallel shift register generators," *IEEE Trans. Commun.*, vol. 45, no. 9, pp. 1053–1060, Sep. 1997.
[6] S. M. Mishra, S. S. Rofail, and K. S. Yeo, "Design of high performance double edge-triggered flip-flops," *Proc. IEE Dev. Syst.* , vol. 147, pp. 283–290, Oct. 2000.