Area-efficient high-throughput parallel scramblers using generalized algorithms

Yun-Ching Tang^{1,2}, JianWei Chen¹, and Hongchin Lin^{1a)}

¹ Department of Electrical Engineering, National Chung Hsing University, Taichung, Taiwan

² Department of Electronics Engineering, Hsiuping Institute of Technology, Taichung, Taiwan

a) hclin@nchu.edu.tw

Abstract: This paper presents generalized algorithms for high-throughput parallel scramblers for digital communication circuits. The proposed algorithm can be applied to any three-term scrambler polynomials with the critical path of one register and one XOR gate using the smallest number of registers. The fan-outs of each register can also be determined by calculation. The test chip reveals that the chip area can be reduced by more than 50% compared with that in the literature, and the power dissipation, including the clock buffers, is only 17.33 mW at 1.6 GHz with 16 parallel outputs, which is equivalent to 25.6 Gbps using TSMC 0.18 μ m CMOS process.

Keywords: high-throughput, parallel scrambler, clock buffer, register

Classification: Electron devices, circuits, and systems

References

- [1] C. Cheng and K. K. Parhi: IEEE Trans. Circuits Syst. II, Exp. Briefs **53** [10] (2006) 1017.
- [2] K. K. Parhi: IEEE Trans. Circuits Syst. I, Reg. Papers **51** [3] (2004) 512.
- [3] S. W. Seetharam, G. J. Minden and J. B. Evans: IEEE Int. Symp. Circuits and Systems **3** (1993) 2011.
- [4] C.-H. Lin, C.-N. Chen, Y.-J. Wang, J.-Y. Hsiao and S.-J. Jou: IEEE Trans. Circuits Syst. II, Exp. Briefs **53** [7] (2006) 558.
- [5] J. Chen, H. Lin and Y.-C. Tang: IEEE Int. Symp. Circuits and Systems (2010) 441.
- [6] H. Lin, Y.-F. Chen and H.-C. She: IEEE Int. Symp. Circuits and Systems **4** (2001) 148.

1 Introduction

Many types of noise affect quality of communications. Single coding technology may not provide an excellent error-correcting capability. To improve the performance, in addition to the Reed-Solomon code, the turbo

code or the low-density parity-check (LDPC) code, scramblers and interleavers are usually adopted. However, serial operation with single bit output may not be feasible for future high-speed transmission, especially in optical communications. Therefore, high-throughput parallel scramblers are required.

Various scrambler polynomials have been adopted. Figure 1 shows the synchronous frame scrambler in the IEEE 802.11a/g/n, which utilizes the modulo-two polynomial $x^7 + x^4 + 1$. The generated pattern is a sequence with a maximum length of 127. For the physical coding sub-layer (PCS) in 802.3ae, the polynomial is $x^{58} + x^{39} + 1$. For the WAN interface sub-layer (WIS), it is $x^7 + x^6 + 1$. In the IEEE 1394 b, the polynomial is $x^{11} + x^9 + 1$. The purposes of these scramblers are to disperse the binary signals with nearly equal zero and one probabilities.

Fig. 1. Synchronous frame scrambler

Pipelining and unfolding have been discussed for parallel processing of linear feedback shift register (LFSR) and cyclic redundancy check (CRC) [1, 2]. Since the algorithms are used for the polynomials with arbitrary terms, they are not optimized for the specific polynomials like scramblers in terms of critical paths, number of registers and fan-outs. The first parallel scrambler was applied to optical communications [3]. It generates parallel outputs by calculating the polynomials using several XOR operations with numerous fan-outs of each register. The work [4] utilizes the look-ahead method with more registers. The critical path is reduced to one XOR and one register. However, for certain polynomials, the algorithm requires numerous registers.

This paper proposes a generalized parallel architecture for any scrambling polynomial. The critical path is minimized to one XOR gate and one register for high speed. Furthermore, it reduces registers and to save chip area and power consumption. To verify the proposed architecture, the dynamic differential circuits were used to design and implement the parallel scrambler for Fig. 1 using the TSMC 0.18 μm CMOS technology. Then, the experimental results and comparison are presented. The final section is the conclusion.

2 Generalized parallel architectures

To derive the generalized parallel scrambler algorithm for the aforementioned various scramblers, the generating polynomial $P(x)$ is assumed to have three terms, given by Eq. (1), where A is the number of shift registers in the conventional serial scrambler and B is the other integer such that $A > B$.

$$P(x) = x^A + x^B + 1 \quad (1)$$

For convenience in the following derivation, D is defined as the difference between A and B . If q_k is the $(k+1)$ 'th generated bit in the serial scrambler, then $Q(j)$, given below, represents the M parallel output sequence in the j 'th cycle of the parallel scrambler.

$$Q(j) = \{q_{M \times j}, q_{M \times j + 1}, \dots, q_{M \times j + M - 2}, q_{M \times j + M - 1}\} \quad (2)$$

For example, consider $P(x) = x^7 + x^4 + 1$; the initial values $\{q_0 \sim q_6\}$ are stored in the registers $\{x^7 \sim x^1\}$, respectively. Since the generating function depends on the generating polynomial $P(x)$, the output bits q_i for $i \geq 7$ can be determined using the following three tips.

1) Looking ahead

For a serial scrambler, the data bit of the next state can be determined using looking ahead [4], which produces the next data bit using the previous state. For example, if the first data bit generated by $P(x)$ is composed of the function $x_7 + x_4$, then the next function is $x_6 + x_3$, and the following is $x_5 + x_2$.

2) Substitution

During looking ahead, if any terms in $P(x)$ are not available, they have to be converted to the other equivalent forms. For example, if the current function is $x_4 + x_1$, then the next becomes $x_3 + x_0 = x_3 + x_7 + x_4$ based on looking ahead and substitution, where x_0 is replaced by $x_7 + x_4$.

3) Merging

For the XOR operation, any two identical terms in the functions can be merged into none due to modulo-2 operation. For instance, if the current function is $x_5 + x_2 + x_1$, then the next function will be $x_4 + x_1 + x_7 + x_4$. Here, x_4 appears twice, so the function can be merged into $x_7 + x_1$.

Based on the above simple tips, the following algorithm of parallel scrambler can minimize the register number and keep the shortest critical path of one register and one XOR gate for high speed and high throughput. With the parameters M , A , B , and $D = A - B$, the following relations can be derived.

A. q_k : output term with index k

For a given k , h is the smallest integer that satisfies the following condition.

$$k < A \cdot 2^h \quad (3)$$

and q_k can be expressed as

$$q_k = q_{k-A \cdot 2^{(h-1)}} + q_{k-B \cdot 2^{(h-1)}} \quad (4)$$

where k must exceeds $A-1$.

B. Register upper bound (R) for the case of M parallel output bits

If the sequence Q from q_0 to q_k , in which $k = R+M-1$, is divided into two parts, then R registers are required to store the data bits that are associated with the first half $\{q_0, q_1, \dots, q_{R-1}\}$; the second half $\{q_R, q_{R+1}, \dots, q_{R+M-1}\}$ represents M parallel output bits. However, from Eq. (4), the index of

$q_{k-B \cdot 2^{(h-1)}}$ must exceed that of $q_{k-A \cdot 2^{(h-1)}}$, so $q_{k-B \cdot 2^{(h-1)}}$ is the condition that limits the number of registers. Substituting the output q_k with the maximum index $k = R+M-1$ into Eq. (4) makes the index of the second term as $R+M-1-B \cdot 2^{(h-1)}$, which is less or equal to $R-1$. Thus, we have $M \leq B \cdot 2^{(h-1)}$. Notably, h may change owing to different k 's, so h for $k = R-1$ is not equal to h for $k = R+M-1$. After selecting a new parameter g equal to $h-1$, we have g to be the smallest integer to satisfy Eq. (5).

$$M \leq B \cdot 2^g \quad (5)$$

Another term that limits the number of registers is the maximum $k = R-1$ for the same g . Substituting the upper limit $k = A \cdot 2^g - 1$ into $k - B \cdot 2^{(g-1)}$ yields $(A+D) \cdot 2^{(g-1)} - 1$, which is equal to $R-1$. Hence, the upper bound (R) is

$$R = (A+D) \cdot 2^{(g-1)} \quad (6)$$

Note that R is the register upper bound for M parallel output bits, but it could be further reduced in some situations to be explained later.

C. Number of fan-outs of each register to XOR gates

After R is determined by Eq. (6) for the given M , the sequence q_R to q_{R+M-1} is divided into two parts, q_R to q_C and q_{C+1} to q_{R+M-1} , in which C is the upper limit $A \cdot 2^g - 1$. The h values for those two groups are g and $g+1$. Using Eq. (4) for q_R and q_C yields $q_R = q_{(A+D) \cdot 2^{(g-1)}} = q_{(A+D) \cdot 2^{(g-1)} - A \cdot 2^{(g-1)}} + q_{(A+D) \cdot 2^{(g-1)} - B \cdot 2^{(g-1)}}$ and $q_C = q_{A \cdot 2^g - 1} = q_{A \cdot 2^g - 1 - A \cdot 2^{(g-1)}} + q_{A \cdot 2^g - 1 - B \cdot 2^{(g-1)}}$. Any k in q_k is constrained within the ranges $(A+D) \cdot 2^{(g-1)} - A \cdot 2^{(g-1)} \leq k \leq A \cdot 2^g - 1 - A \cdot 2^{(g-1)}$ and $(A+D) \cdot 2^{(g-1)} - B \cdot 2^{(g-1)} \leq k \leq A \cdot 2^g - 1 - B \cdot 2^{(g-1)}$. Those are

$$D \cdot 2^{(g-1)} \leq k < A \cdot 2^{(g-1)} \quad (7.1)$$

$$D \cdot 2^g \leq k < (A+D) \cdot 2^{(g-1)} \quad (7.2)$$

Similarly, q_{C+1} and q_{R+M-1} are expressed as $q_{C+1} = q_{A \cdot 2^g} = q_{A \cdot 2^g - A \cdot 2^g} + q_{A \cdot 2^g - B \cdot 2^g} = q_0 + q_{D \cdot 2^g}$ and $q_{R+M-1} = q_{(A+D) \cdot 2^{(g-1)} + M - 1} = q_{(A+D) \cdot 2^{(g-1)} + M - 1 - A \cdot 2^g} + q_{(A+D) \cdot 2^{(g-1)} + M - 1 - B \cdot 2^g} = q_{M - B \cdot 2^{(g-1)}} + q_{M - D \cdot 2^g - B \cdot 2^{(g-1)}}$. Thus, the other two relations are

$$0 \leq k < M - B \cdot 2^{(g-1)} \quad (7.3)$$

$$D \cdot 2^g \leq k < M + D \cdot 2^g - B \cdot 2^{(g-1)} \quad (7.4)$$

The fan-outs of each register can be determined in two steps. Step 1 is based on Eqs. (7.1) to (7.4). If k satisfies one, two, three, or all of these four equations, then register k has one, two, three, or four fan-outs to the XOR gates, respectively.

In Step 2, we consider whether the outputs of registers feed back to another registers by searching fan-outs of register k from $k = 0$ to $R - M - 1$. If yes, the fan-outs of register $k+M$ are increased by one. Note that the fan-outs of registers could be zero, which indicates the registers could be removed after all of the feedback paths of the registers are checked.

D. Minimum number of registers

Based on the above algorithm, two parallel scrambler architectures, register-last and register-first, can be constructed and illustrated in Figs. 2 and 3. In Fig. 2, the feedback data enter the XOR gates, and the registers (REG) are placed after the XOR's. The $REG \times M$ register block is composed of M registers with M XOR gates in front of them. Since each XOR can be merged with one register as indicated by dashed lines for implementation, that facilitates the design of a special register with an XOR function [4, 5]. The other register block has $R_{ml} - M$ registers without XOR before them.

Fig. 2. Architecture of register-last parallel scrambler

In Fig. 3, the feedback data enter the registers, and then go to the XOR gates. Some of the XOR gates may have one more fan-out, so some registers may have one less fan-out.

Fig. 3. Architecture of register-first parallel scrambler

The minimum numbers of registers (R_{ml} and R_{mf}) in the register-last and register-first parallel scramblers can be determined by

Register – last : for $0 \leq k \leq R - M - 1$

$$R_{ml} = \text{number of nonzero fanouts} + M$$

Register – first : for $0 \leq k \leq R - 1$

$$R_{mf} = \text{number of nonzero fanouts}$$

Table I gives several scrambler polynomials are chosen as the examples for given k with $M = 8$ and $M = 16$ parallel outputs. Table II lists the examples of three polynomials with the parameters M , R , R_{mf} and R_{ml} . The upper bound R on the number of registers is determined first. From the fan-outs of all registers, the minimum number of registers can be determined.

Table I. Examples to output q_k

Polynomial	k	M	h	g	q_k
$x^7 + x^4 + 1$	23	8	2	1	$q_9 + q_{15}$
	31	16	3	2	$q_3 + q_{15}$
$x^7 + x^6 + 1$	23	8	2	1	$q_9 + q_{11}$
	31	16	3	2	$q_3 + q_7$
$x^9 + x^7 + 1$	23	8	2	1	$q_5 + q_9$
	31	16	2	2	$q_{13} + q_{17}$

Table II. Examples to find R , R_{mf} , and R_{ml} for the different polynomials

Polynomial	M	R	R_{mf}	R_{ml}
x^7+x^4+1	16	20	20	20
x^7+x^2+1	8	24	18	18
x^7+x^3+1	10	22	16	18

3 Comparison and analysis

Table III compares the number of registers, the maximum fan-outs of registers and XOR gates between the proposed architectures and those in the works [4] for two scrambler polynomials with 16 parallel outputs. All of them have 16 2-input XOR gates with the critical paths of one register and one XOR gate. Lin, *et al.* [4] developed a systematic parallel scrambler design methodology, but more registers than those of our proposed architectures are needed when $A-B$ in Eq. (1) is getting larger. Our register-last and register-first algorithms are denoted as RL and RF, respectively. The proposed register-first scrambler can reduce the number of registers by more than 40%.

Table III. Comparison of architectures for $M=16$

Polynomial	Algorithm	Registers	Fan-outs (register)	Fan-outs (XOR)
$x^7 + x^4 + 1$	Ref. [4]	28	3	1
	RL	20	4	1
	RF	20	3	2
$x^7 + x^3 + 1$	Ref. [4]	48	4	1
	RL	32	4	1
	RF	28	3	2

For some polynomial such as x^7+x^6+1 , the proposed algorithm has the architecture identical to that described elsewhere [4]. However, for the polynomial x^7+x^4+1 , the proposed schemes utilize fewer registers. Table IV gives the gate-level comparison of the proposed architecture and that in the cited work [4] for 16 parallel outputs. Both methods have the same critical path that consists of one register plus a two-input XOR operation. The proposed architecture has fewer registers. Even though some registers may have one more fan-out than, its influence on speed should be insignificant.

Table IV. Comparison of architectures for $M=16$

	Proposed	Ref. [4]
Polynomial	$x^7 + x^4 + 1$	
No. of registers	20	28
Critical path	One register with 2-input XOR	
Max. fan-outs	3	2

4 Circuit design for the parallel scrambler

To design the parallel scrambler using the proposed algorithm, only two logic blocks are required. One is a register. Another is a register with a two-input XOR. Since these two logic blocks perform mutually symmetrical operations, differential logic [6] was adopted for transistor-level design. Here, the modified dynamic differential logic gates are used to enhance speed and reduce power.

Figure 4 shows the differential logic gates for a register and a register with a two-input XOR, where the tri-state inverter is illustrated in Fig. 4 (c). A and B are the inputs, while OUT is the output. Figure 5 illustrates the 3-phase clock patterns, CLK_1 , CLK_2 and CLK_3 , used in the circuits. One clock cycle is divided into six small time slots t_1 to t_6 . During t_1 , since CLK_1 is high to turn on the two NMOS switches with their gates tied to CLK_1 , the signals will be passed to the two PMOS cross-coupled nodes. At t_2 , CLK_3 goes low to switch the PMOS with its gate tied to CLK_3 to pull one of the two cross-coupled nodes to V_{DD} . The time t_3 is a “buffering” time to avoid the signals leaked from the inputs to the outputs. During t_4 , CLK_2 becomes high to activate the tri-state inverters. The data are transferred to the outputs and latched. Next, CLK_3 is high to turn off the PMOS at t_5 , which could be short for preparation to receive the next data. Finally, t_6 is also short like t_3 . Since all of the clocked controlled transistors are off, the circuit is ready for the next data bit

Fig. 4. (a) The differential register (b) The differential register with 2-input XOR (c) The tri-state inverter

Fig. 5. The clock patterns of CLK₁, CLK₂, and CLK₃

5 Implementation and results

Figure 6 illustrates the proposed parallel scrambler for $x^7 + x^4 + 1$ with $M = 16$, where P_0 to P_{15} refer to the output nodes. The blocks of R and X-R correspond to the circuits in Figs. 4 (a) and 4 (b), respectively. It was designed and fabricated using TSMC 0.18 μm CMOS technology in the

Fig. 6. The proposed parallel scrambler of $x^7 + x^4 + 1$ with $M = 16$

(a) Layout

(b) Chip microphotograph

Fig. 7. The top view of proposed parallel scrambler

Fig. 8. The measured waveform of the output signal toggled every 127 clocks

area of $95\mu\text{m} \times 70\mu\text{m}$. Figure 7 (a) shows the layout of the parallel scrambler, which corresponds to the block marked by the red lines in Fig. 7 (b). The rest of the chip contains the clock generator, detection circuit, clock buffers, I/O buffers and power rings. A test circuit was also designed to detect the accuracy of the parallel scrambler by designing a detection circuit to reverse the output signal if it is matched after each 127 cycles. Figure 8 shows the measured waveform, in which the output signal is reversed with the period of 158.75 ns at a clock frequency of 1.6 GHz. The power consumption of the parallel scrambler (including the clock buffers) is 17.33 mW.

Table V compares the implementation results of the proposed parallel scrambler and Ref. [4]. The proposed circuit has higher throughput per unit area and lower power consumption divided by throughput. Therefore, the proposed differential logic circuit is area-efficient, while the double-edge triggered registers used in the work [4] may have higher parasitic effects.

Table V. Performance comparison of different architectures after implementation

Architecture	Proposed	Ref. [4]
Technology	CMOS 0.18 μm	CMOS 0.18 μm
Function	$x^7 + x^4 + 1$	$x^7 + x^6 + 1$
Number of parallel outputs	16	16
Number of register	20	16
Max. fan-outs	3	3
Frequency (GHz)	1.6	1.25 (double-edge)
Throughput (Gbps)	25.6	40
Power consumption (mW)	17.33	40.49
Area (μm^2)	95×70	150×90
Power/ Throughput (mW/Gbps)	0.68	1.01
Throughput/Area (Mbps/ μm^2)	3.85	2.96

6 Conclusion

The generalized systematic parallel scrambler algorithm and architecture with the critical path of one register and one XOR gate are developed with the minimum register number for high throughput. The proposed algorithm can be applied to any scrambler polynomials with three terms to achieve small numbers of registers and fan-outs. Specially, the proposed dynamic differential registers for the polynomial $x^7 + x^4 + 1$ with $M = 16$ meet the requirement of high speed, high throughput, low power and small chip area. The results reveal that the power dissipation, including that of the clock buffers is 17.33 mW at 1.6 GHz with a throughput of 25.6 Gbps in a chip area of $95\mu\text{m} \times 70\mu\text{m}$ using TSMC 0.18 μm CMOS technology. The throughput per unit area is up to 3.85 Mbps/ μm^2 .

Acknowledgments

The authors would like to acknowledge the Chip Implementation Center (CIC) of the National Applied Research Laboratories of Taiwan for the support in chip fabrication. This work was supported by National Science Council of Taiwan (NSC 101-2220-E-005-013) and was supported in part by the Ministry of Education, Taiwan under the ATU plan.