www.czasopisma.pan.pl PAN POLSKA AKADEMIA NAUK www.journals.pan.pl

INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2018, VOL. 64, NO. 1, PP. 91-94
Manuscript received August 30, 2017; revised January, 2018. DOI: 10.24425/118151

The Implementation of the Parallel Scrambler Scheme for the IEEE 802.11 Standard

Alexey Kudinov, Yaroslav Antimirov, Igor Tyshchenko, Mariia Popova, and Alexander Cherepanov

***Abstract*—This article is devoted to the development of the scrambler circuit. Nowadays, new WiFi standard IEEE 802.11 is being put into operation, so that there is a huge need in modern, energy-efficient algorithms, which will be used in the data transmission. Consequently, some of the scrambler circuits, which could be implemented for the IEEE 802.11 standard are described with its comparison. In addition, an example in Python is given for readers to use it in their researches.**

***Keywords*— scrambler, sequential circuit, parallel circuit.**

I. Introduction

IN digital wireless communication systems, scrambling of the transmitted digital data stream is often used. This operation is performed to exclude long sequences of zeros and ones from the transmitted data stream, which has a beneficial effect on the energy characteristics of the transmission channel [1].

At present, when building systems, much attention is paid to their energy efficiency, which forces their developers to seek ways to reduce the power consumed by digital processing units. Despite the progress in the optimization algorithms implemented in the logic synthesis tools of digital circuits, the structure of the original description of the circuit in high-level languages (VHDL or Verilog) has a significant effect on the parameters of the circuit obtained from such a description.

We will consider the options for building a scrambler scheme, designed to work as part of a micro-consuming digital data transmission system. Accordingly, one of the main criteria for estimating the scheme was the power consumed by it.

II. Sequential Circuit

Figure 1 shows a block diagram of the scrambler used in the IEEE 802.11 wireless data transmission standards family [2].

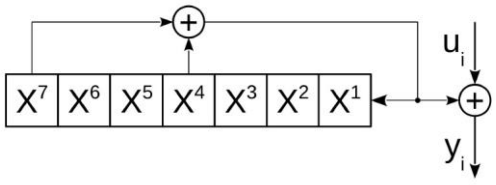

Fig. 1. Scrambler scheme used in the data transfer protocol family IEEE 802.11.

The Russian Ministry of Education and Science funded the works on RFMEFI57815X0136 under the agreement №14.578.21.0136 of October 27 2015.

Authors are with Ural Federal University named after the first president of Russia Boris Yeltsin, Yekaterinburg, Russia (e-mail: alexey.kudinov@urfu.ru).

The basis of this scrambler is a 7-bit shift register X. Outputs of cells X^4 and X^7 of the shift register are summed modulo 2 and fed back to its input, thus the bit sequence generator is made by the generating polynomial:

$$S(x) = x^7 + x^4 + 1 \quad (1)$$

The next (i-th) bit of this sequence is summed modulo 2 with a bit of input data u_i, forming the next bit y_i of the scrambled sequence.

Since in this form the scrambler circuit processes all data bits in series, then, at significant information exchange speeds, it will become necessary to organize the operation of this circuit at sufficiently high frequencies, which may be undesirable or even impossible. Therefore, in many cases it is advisable to go from sequential processing of the data stream to a parallel group of w bits. Next, we consider several ways to implement a parallel form of the scrambler.

III. "Combinatorial" Parallel Circuit

As it is known, the state of a discrete time linear system can be described as follows: [3, 4]

$$\begin{cases} X(i+1) = FX(i) + GU(i) \\ Y(i) = HX(i) + JU(i) \end{cases} \quad (2)$$

where X is the system state, U is the input, and Y is the system output.

One possible way to implement the "parallel" form of the scrambler is shown in Fig. 2.

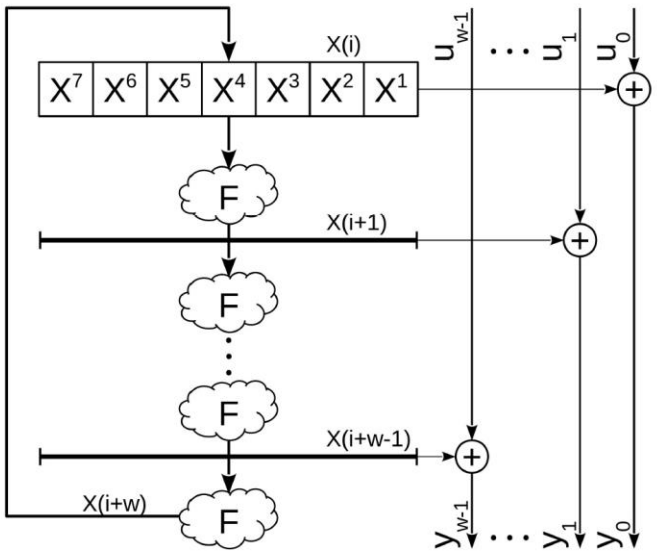

Fig. 2. "Combinatorial" parallel scrambler circuit.

Here, from the initial state $X(i)$ stored in the register, using a chain of identical combinatorial circuits, a series of states $X(i+1)$, $X(i+2)$, … $X(i+w-1)$ is generated, the lowest bits of which are taken for addition modulo 2 with bits u_0, u_1, … u_{w-1} and obtaining a set of output bits y_0, y_1, … y_{w-1}. The last combinatorial transformation generates a state $X(i+w)$ that is stored in register and is the initial one for scrambling the next w-bit data set.

An obvious drawback of such a scheme is the formation of combinatorial chains, the length of which increases rapidly with the increase in the width of the input data bus, which limits the maximum achievable operating frequency of the circuit, and, as can be easily seen, introduces a noticeable combinatorial delay in the propagation of data from the input to the output of the circuit.

IV. "MATRIX" PARALLEL CIRCUIT

Another implementation of the parallel scrambler scheme can be synthesized using the approach described in [5].

It is known [5, 6] that the set of values of 0 and 1 with operations of logical multiplication · ("AND") and addition of modulo 2 ⊕ ("exclusive OR") form a Galois field:

$$\left\{\{0,1\},\bigoplus,\cdot\right\} = GF(2) \tag{3}$$

Thus, in the calculation of matrices, one can replace the operations of scalar multiplication and addition by operations of logical multiplication and addition modulo 2, respectively; and the matrix multiplication operation can be realized using a circuit containing the «AND» and «XOR» logical elements. Then, in the case of the sequential circuit depicted in Fig. 1, $U(i)$, $Y(i)$ and J respectively are scalars (scalar quantities are denoted by small letters $u(i)$, $y(i)$ and j); G, H and X are columns with a height of 7 elements, and F is a square matrix with 7×7 size. Then these elements have the following values:

$$\begin{matrix} G = 0 \\ j = 1 \end{matrix} \tag{4}$$

$$H = (0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1) \tag{5}$$

$$F = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{6}$$

From (2) it is not difficult to calculate the state of the system at any step, for example:

$$X(i+w) = F^w X(i) \tag{7}$$

We get the scheme of the parallel scrambler, depicted in Fig. 3. The main task in the construction of this scheme is the synthesis of the combinatorial region that realizes the matrix multiplication, for which it is necessary to calculate the matrix F^w. The iteration method is most straightforward:

$$F^{i+1} = FF^i \tag{3}$$

Analysing equations (4) and (5), it is easy to see that the matrix F has the following structure:

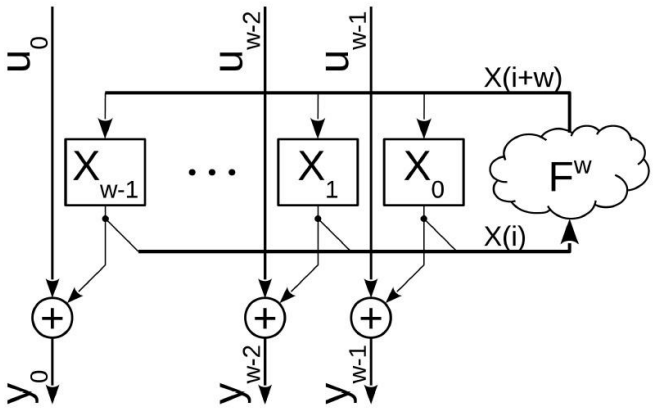

Fig. 3. "Matrix" parallel scrambler circuit.

$$F = \left(\begin{array}{c|c} \multicolumn{2}{c}{H} \\ \hline I_{w-1} & \begin{matrix}0\\0\\\vdots\\0\end{matrix} \end{array}\right) \tag{9}$$

So that we have:

$$\left(\begin{array}{c} HF^i \\ \hline F^i \end{array}\right) \tag{10}$$

where F^i has first m-1 rows.

Thus, having a generating polynomial S (1), it is possible to construct the original matrices H (5) and F (6), and then, successively applying (10), to obtain the desired transformation matrix F^w. Having a transformation matrix it is easy to synthesize the circuit implementation.

It should also be noted that in cases where the width of the input word is greater than the degree of the generating polynomial, the vector H (5) must be complemented by the corresponding number of zeros on the right (which is equivalent to adding the triggers to the shift register shown in Fig. 1, to the left).

To generate a combinatorial scheme that computes $X(i-w)$ from $X(i)$, a generator was written in Python. The circuit description is generated in Verilog language.

Generator in Python:

```
import numpy

t  = numpy.dtype('i1')
# Datapath width
w  = 64
#The generator polynomial
# S(x) = x^7 + x^4 + 1
S  = numpy.asarray([0, 0, 0, 1, 0, 0, 1, 0], t)
lS  = len(S)
# Matrix H derived from the generator polynomial by extending with zeroes
zH  = numpy.zeros((w - lS,),t)
H  = numpy.concatenate((S,zH))
# Zero column for F matrix construction
Z  = numpy.zeros((w-1,1), t)
# Identity matrix for F matrix construction
I  = numpy.identity(w-1, t)
```

```
# Assemble the F matrix
F0  = numpy.hstack((I,Z))
F   = numpy.vstack((H,F0))

F1  = numpy.empty((w,w), t)
print(F)

# Calculate the F^w matrix iteratively
for it in xrange (1, w) :
  print('F^%d:' % (it+1))
  for j in xrange (1, w) :
    F1[j,:] = F[(j-1),:]

  for i in xrange (0, w) :
    v = 0
    for j in xrange (0, w-1) :
      v = v ^ (H[j] & F[j,i])
    F1[(0,i)] = v
  numpy.copyto(F, F1, casting='no')
  print(F)

print ('Verilog:')
for i in xrange (0, w) :
  l = ['X%d[%2d] =' % (w,i)]
  v = F[i,:]
  c = 0
  for j in xrange (0, w) :
    if v[j] == 1 :
      if c > 0:
        l.append('^');
      l.append('X[%2d]' % (j))
      c = c+1

  s = ' '.join(l) + ';'
  print(s)

# Test
print ('Test:')
x   = numpy.ones(w, t)
x1  = numpy.empty(w, t)

for it in xrange (0, 10) :
  l   = []
  cnt = 0
  for i in xrange (0, w) :
    v = F[i,:]
    xi= 0
    for j in xrange (0, w) :
      if v[j] == 1 :
        xi = xi ^ x[j]
    x1[i] = xi
    l.insert(0, '%d' % xi)
    cnt = cnt + 1
    if cnt == 8 :
      l.insert(0, ' ')
      cnt = 0
  numpy.copyto(x, x1, casting='no')
  print ('%2d: ' % (it) + ''.join(l))
```

V. Experimental results

To evaluate the solutions obtained in different ways, a synthesis of the scrambler unit with a bit width of the input bus of 64 bits in the basis of standard cells with a supply voltage of 1.8 V of the 180 nm process (CL180G) of the TSMC factory was carried out. To assess the approaches to implementing the scrambler described in this article, three versions of the description in the language of System Verilog were prepared:

1. block with a long combinatorial structure (Fig. 2), with a register at the output;
2. block with the proposed matrix structure (Fig. 3);
3. block with a long combinatorial structure (Fig. 2), without a register at the output;
4. block with the proposed matrix structure (Fig. 3), in which the register stores only the lowest 7 status bits, and all other bits are computed combinatorically.

TABLE I
Parameters

	Slow	**Med**	**Fast**
τ, ns	2.00	1.50	1.00
Δ τ, ns	0.20	0.15	0.10

The presence of an output register in the block makes it possible to isolate the delay created by combinatorial circuits, but introduces a "synchronous" delay of 1 clock cycle.

For the synthesis, three sets of time parameter constraints were prepared:
where τ is the period of the clock signal, and Δτ is the instability of the edges of the clock signal.

The main parameters of the circuits obtained as a result of the synthesis of four variants of scrambler circuits are given in Table II.

Analysing the data shown in the table, we can note the following:

1. The achieved parameters of circuit variants, in the Verilog-descriptions of which there are long combinatorial chains (variants 1 and 3), are close to the parameters of the circuit, in the description of which these chains are much shorter (variants 2 and 4), which indicates a very good degree of optimization of combinatorial circuit which is executed automatically at the stage of logical synthesis;
2. The time parameters achieved in the "Med" and "Fast" sets are the limits for these schemes;
3. All four variants are workable only in conditions "Slow", i.e. clock frequency is about 500 MHz;
4. "Combinatorial" scheme with output register (option 1) is the worst in all parameters;
5. The proposed "matrix" structure with a reduced status register (option 4) is most preferable in terms of the occupied area of the crystal and the power consumption in cases where the speed requirements are relatively low. However, this version has a delay in the distribution of data before the release, which can not be optimized and is 1.5 ns;
6. The proposed "matrix" structure with a full state register (option 2) introduces a smaller delay in the distribution of data from the input to the output as compared to options 3 and 4, but with greater power consumption and a larger crystal area.

TABLE II
SYNTHESIS RESULTS

	Slow				Med				Fast			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
Synthesis time, s	11	8	8	10	60	36	65	122	60	58	128	113
Area, μm^2	8056	7420	5679	5556	11755	10721	11908	13083	16225	12368	14572	12464
Power, µW	16086	14801	11360	10681	26484	24253	23478	24136	47798	34723	37903	32392
Number of triggers	71	63	7	7	71	63	7	7	71	63	7	7
Number of valves	253	243	234	215	320	338	465	487	489	349	596	476
Δt C2C, ps	1716 (1717)	1694 (1696)	1447 (1697)	1386 (1660)	1401 (1257)	1367 (1259)	1236 (1237)	1276 (1256)	1433 (827)	1350 (770)	1188 (808)	1193 (782)
Δt I2O, ps	-	673 (1800)	1043 (1800)	1020 (1800)	-	659 (1350)	1026 (1350)	954 (1350)	-	703 (900)	924 (900)	900 (900)
Δt C2O, ps	-	1707 (1800)	1800 (1800)	1799 (1800)	-	1349 (1350)	1586 (1350)	1466 (1350)	-	1133 (900)	1595 (900)	1616 (900)

where Δt C2C - signal passage from the synchronous element to the synchronous element (clock-to-clock); Δt I2O - the signal propagation from the input of the circuit to its output (input-to-output), in the first variant of the circuit the input data passes through the register, because of which the estimation of I2O has no meaning for it; Δt C2O - the delay of the passage of the signal from the synchronous element to the output of the circuit. In parentheses there are the maximum permissible values calculated by the synthesizer taking into account the period of the clock signal, the instability of its fronts, the time for presetting the signals at the inputs of synchronous elements, and so on. When calculating the block area, only the area occupied by the logical cells was taken into account.

VI. CONCLUSION

There is a huge need in modern, energy-efficient algorithms, which could be implemented for new standards of WiFi, proposed in the beginning of 2017. In this article some scrambler circuits, which are widely used in data trancmission protocols are described with the example in Verilog.

The proposed variant of the IEEE 802.11 scrambler scheme construction, based on matrix transformations, allows, depending on the requirements, to synthesize more efficient or more rapid variants of parallel schemes, which could be used as a part of a transmit-receive systems in the various types of industries with the development of Internet of Everything (from the monitoring of the appliances in factories to the usage in medicine).

In addition, there is a need in a future work, as it is important to check if the proposed algorithm works well in a various circumstances and to improve it further.

REFERENCES

[1] W. Stallings, “Data and Computer Communications, 8th Edition”. Upper Saddle River: Pearson Prentice Hall, 2007 pp. 146-151.

[2] 802.11ah-2016 - IEEE Standard for Information technology--Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 2: Sub 1 GHz License Exempt Operation

[3] Lotfi A. Zadeh and Charles A. Desoer, “Linear System Theory: The State Space Approach”. N.Y.: McGraw-Hill Book Company, 1963, 628 p.

[4] G. Albertengo, R. Sisto, “Parallel CRC Generation,” *IEEE Micro*, vol. 10, no. 5, pp. 63-71, Oct. 1990. DOI: 10.1109/40.60527

[5] Giuseppe Campobello, Giuseppe Patane, Marco Russo, “Parallel CRC Realization,” *IEEE Transactions on Computers*, vol. 52, no. 10, pp. 1312-1319, Oct. 2003. DOI: 10.1109/TC.2003.1234528

[6] J.Borges and J.Rifa, “A Characterization of 1-Perfect Additive Codes,” *IEEE Transactions on Information Theory*, vol. 45, no. 5, pp. 1688 - 1697, Jul. 1999. DOI: 10.1109/18.771247