Resource Utilization of Stream Cipher Systems in Reconfigurable Hardware

V A Moorthi Paramasivam

Centre for Artificial Intelligence and Robotics, Rajbhavan Circle, Bangalore 560001, Karnataka, INDIA, moorthi@cair.res.in

Amshuman Bhardwaj Preethi Tibrewala Shilpa Garde

Dept of Telecommunication Engineering, VTU, RVCE, Bangalore 560001. Karnataka, INDIA

Abstract

High throughput and high-speed encryption and decryption are the most wanted criteria for present day secure communication requirement. Increasingly, security standards and applications are defined to be algorithm independent. Although context switching between algorithms can be easily realized via software implementations, the task is significantly more difficult when using classical hardware implementations. In this work we have studied some of the pseudorandom sequence generators and their hardware resource utilization in reconfigurable hardware (FPGAs) .

Keywords: Linear Feedback Shift Register, Psuedo Random Sequence Generators (PRSG), Statistical Tests

1 INTRODUCTION

Stream ciphers are an important class of encryption algorithms. They encrypt individual characters (usually binary digits) of a plaintext message one at a time, using an encryption transformation, which varies with time. Stream ciphers can be very fast to operate; they are generally much faster than block ciphers. Since the keystream can often be generated independently of the plaintext or ciphertext such generators often have the advantage that the keystream can be generated prior to encryption or decryption, with only an easy combining step left when the message or ciphertext is to be processed.

In general stream ciphers use Linear Feed Back Shift Registers to generate pseudo random sequences (Keystream if the statistical properties of meets as in ch. 5.4 of [8]). To generate the Keystream LFSRs are not used as it is ; but they used with some combinational logic to introduce nonlinear characteristics in the sequences generated by them.

In section 2, we brief out the types of pseudorandom bit generators, which are used in the cryptographic systems. Section 3 details about the realization in FPGA and the hardware realization of some of the generators outlined in section 2. In this paper we concentrated in Step Control and Multiclocked Generators. All these generators are verified for their pseudo randomness as explained in [1]. Section 4 explains about the tool and FPGA used in realization of these generators and section 5 details about the further improvements and future work that can be carried out.

2 PSEUDORANDOM BIT GENERATORS

The Pseudo Random Bit Generators can be classified in to:

1) MULTIPLEXED (Non Linear Feed Forward Transformation)
2) STEP CONTROL
3) MULTICLOCKED

In this work we designed and implemented the following generators and observed their logic cell utilization in FPGA.

1. MULTIPLEXED (Non Linear Feed Forward Transformation)
   i) Geffe Generator
   ii) Jennings Generator
2. STEP CONTROL
   i) Bilateral Step Control
   ii) Gollman Cascade Stop-and-Go Generator
   iii) Beth Piper Stop and Go Generator
3. MULTICLOCKED
   i) The Massey Rueppel’s Multispeed Generator
   ii) Ruppeel’s Self Decimator
   iii) Gollman’s Self Decimator

Logical Block diagrams of the above generators and their properties are available in [1]. In this paper the hardware block diagrams are discussed than the logical block diagrams

3 HARDWARE REALIZATION AND RESOURCE UTILISATION

3.1 System Design

The pseudorandom sequence generators were incorporated into two blocks: the Encryptor and the Decryptor and these two blocks were cascaded to form a stream cipher system as shown in figure 3.1.

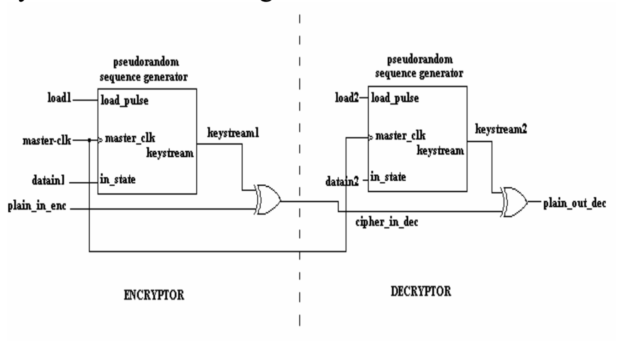


Figure3.1 The stream cipher system in FPGA.

In this work all the pseudo random sequence generators are realized in real time reconfigurable hardware (FPGAs). The hardware utilization in this result includes the hardware consumption of loading of initial stages of LFSRs, clock generation circuits for Generators which are using more than one clocks , and basic combinational circuits required for these circuits. In this work we concentrated in Step Control and Multi clocked generators, since their statistical properties meet the criteria of pseudo randomness.

In this work we used the configuration as given below to realize the generators in FPGA.

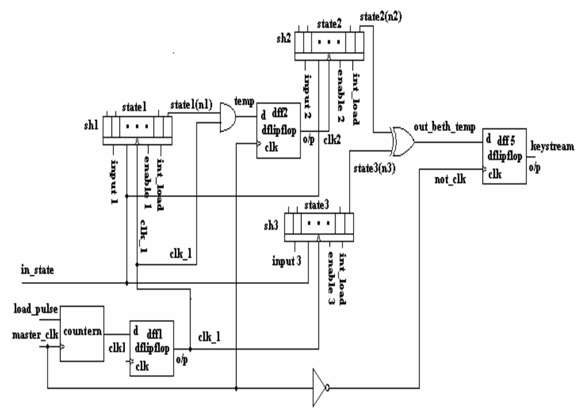
Target FPGA family	*: FLEX 6000*
Target Device	*: EPF6016TC144-3*
Vendor name	*: Altera*
Package used for compilation and pin assignment	*: Quartus 2.0*
HDL used	*: VHDL*
Type of device	*: FPGA*
Target Device Operating Frequency	*: 36 Mhz*
Voltage Levels For the device	*: 5.0V*
Typical number of gates in the target device	*: 16000*
Number of logic elements in the target device	*: 1320*
Maximum I/O pins in the target device	*: 204*
Synthesis optimization	*: Optimized for Speed*

3.2 Resource Utilization of Pseudo Random sequence generators in FPGA

3.2.1 Step Control Generators

3.2.1.1 Beth Piper[5]

This generator uses the output of one LFSR to control the clock of another LFSR as shown in the figure 3.4. The clock input of LFSR-2 is controlled by the output of LFSR-1 so that, LFSR-2 can change its state at time *t* only if the output of LFSR-1 was 1 at time *t-1*. In order to guarantee a good statistical behavior of the stop-and-go sequence, it is suggested that the output sequence of LFSR-2 is XOR-ed with another *pn* sequence (i.e. of LFSR-3)



3.2 Schematic Diagram of the Beth Piper Generator

Beth-Piper Generator

S. No	**Combination**			**Logic element utilization**	
	N_1	**n_2**	**n_3**	**No of cells used out of 1320**	**Percentage utilization**
1.	**5**	**7**	**11**	**90**	**6**
2.	**6**	**8**	**12**	**95**	**7**
3.	**7**	**11**	**13**	**95**	**7**
4.	**8**	**12**	**14**	**107**	**8**
5.	**13**	**17**	**19**	**122**	**9**
6.	**14**	**18**	**20**	**127**	**9**
7.	**17**	**23**	**29**	**143**	**10**
8.	**18**	**24**	**30**	**150**	**11**
9.	**31**	**37**	**41**	**186**	**14**
10.	**32**	**38**	**42**	**195**	**14**
11.	**41**	**43**	**47**	**212**	**16**
12.	**42**	**44**	**48**	**219**	**16**
13.	**59**	**61**	**67**	**224**	**20**
14.	**60**	**62**	**68**	**283**	**21**

Table 3.1 LC utilization of Beth Piper Generator

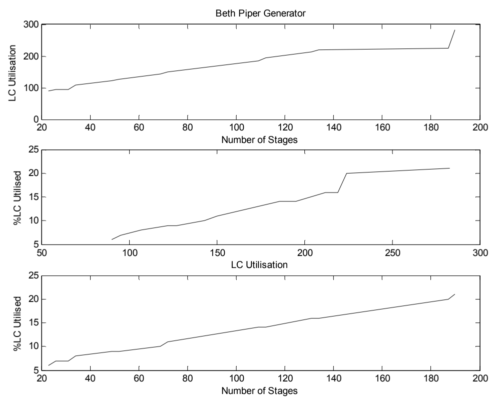
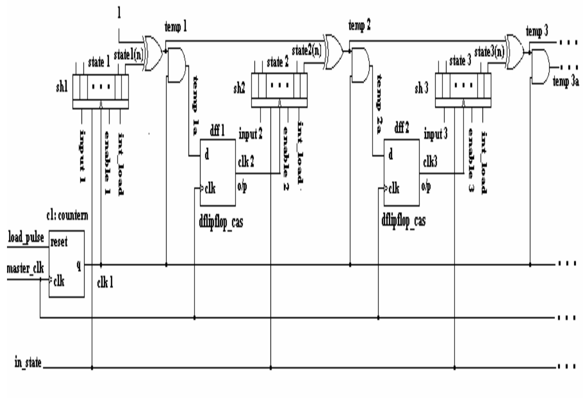


Figure 3.3 LC utilization of Beth Piper

3.2.1.2 The Gollman Cascade Stop-and-Go Generator[1]

Figure 3.5 shows the Gollman cascade generator. It is a strengthened version of the Beth Piper Generator. It consists of a series of l LFSRs with primitive feedback polynomials of the same degree n. LFSR$_i$ is clock controlled jointly by all LFSR$_{j,}$ $j < i$, in the same way as in the Beth Piper scheme.

This generator consists of a series of LFSRs, with the clock of each controlled by the previous LFSR. If the output of LFSR-1 is 1 at time *t-1*, then LFSR-2 is clocked. If the output of LFSR-2 is 1 at time *t-1*, then LFSR-3 is clocked and so on. The output of the final LFSR is the output of the generator. Cascades are conceptually very simple and they can be used to generate sequences with large periods, large linear complexities and good statistical properties.



3.4 Schematic Diagram of the Gollman Cascade Stop-and-Go Generator

Gollman Cascade

S. No	**Combination**		**Logic element utilization**	
	n_lfsr	**N**	**No of cells used out of 1320**	**Percentage utilization**
1.	**3**	**5**	**78**	**5**
2.	**3**	**6**	**87**	**6**
3.	**3**	**7**	**87**	**6**
4.	**3**	**8**	**92**	**6**
5.	**3**	**13**	**112**	**8**
6.	**3**	**14**	**117**	**8**
7.	**3**	**17**	**122**	**9**
8.	**3**	**18**	**131**	**9**
9.	**3**	**31**	**105**	**12**
10.	**3**	**32**	**171**	**12**
11.	**3**	**41**	**204**	**15**
12.	**3**	**42**	**213**	**16**
13.	**3**	**59**	**264**	**20**
14.	**3**	**60**	**266**	**20**

Table 3.2 LC utilization of Gollman Cascade Generator

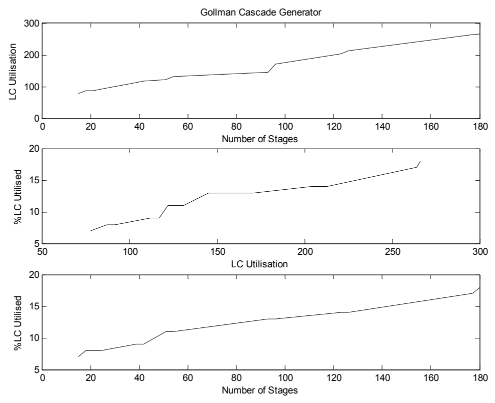


Figure 3.5 LC utilization of Gollman Cascade Generator

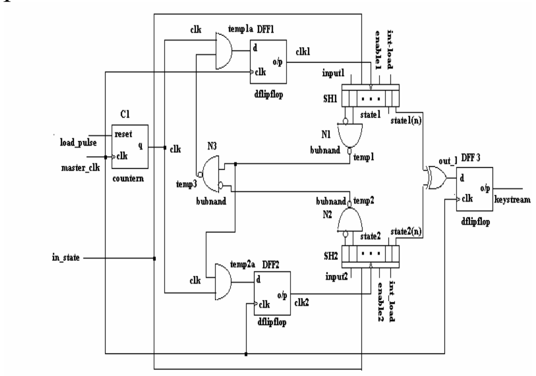
3.2.1.3 The Bilateral Stop-and-Go Generator[1]

This scheme is based on the observation that if a binary sequence b has an odd prime period T, its linear complexity $LC(b)$ will be bound from below by the order $Ord_T(2)$ of the number 2 modulo T, that is $LC(b) \geq Ord_T(2)$.

A sequence with a prime period is desirable because one can subject it to various further cryptographic transformations without influencing the established lower bound to its linear complexity, provided the transformations do not render it into a constant sequence. However, if $T = 2^n - 1$ happens to be one of the Mersenne primes, then we shall have $Ord_T(2)$=n. So T should be chosen to be a prime beyond the progression $2^n - 1$.Sequences with periods of the form $q.\ 2^n - 1$ with odd $q \geq 3$ have been constructed from a pair of n-stage LFSRs.

Figure 3.5 shows the logic diagram for the Bilateral Generator. Here both the LFSRs are started in some non-zero states. Step control is carried out in the following manner:

1) If *(a(t + n – 1), a(t + n –2)) = (0,1)*, then the clock pulse to LFSR-B is to be blocked.
2) If *(b(t + n – 1), b(t + n – 2)) = (0,1)*, but *(a(t + n – 1), a(t + n –2)) ≠ (0,1)*, then the clock
pulse to LFSR-A is blocked.



3.6 Schematic Diagram of the Bilateral Stop-and-Go Generator

Bilateral Stop and Go Generator

S. No	Combination		Logic element utilization	
	n_1	n_2	No of cells used out of 1320	Percentage utilization
1.	11	11	105	7
2.	12	12	109	8
3.	13	13	112	8
4.	14	14	114	8
5.	19	19	127	9
6.	20	20	127	9
7.	29	29	148	11
8.	30	30	152	11
9.	41	41	176	13
10.	42	42	182	13
11.	47	47	187	14
12.	48	48	194	14
13.	67	67	236	17
14.	68	68	244	18

Table 3.3 LC utilization of Bilateral Stop and Go Generator

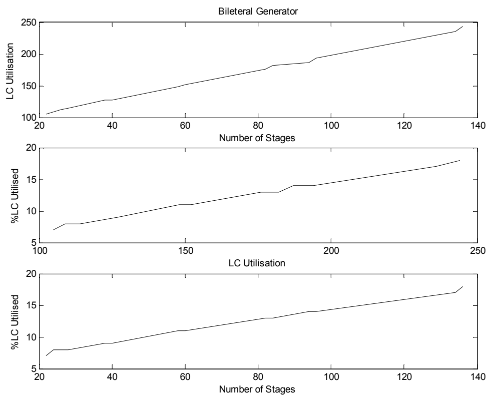


Figure 3.7 LC utilization of Bilateral Stop and Go Generator

3.2.2 Multiclocked Generators

3.2.2.1 The Massey Rueppel's Multispeed Generator[1]

This generator uses two LFSRs clocked at different speeds as shown in figure 3.6. LFSR-2 is clocked *d* times as fast as LFSR-1. The individual bits of the two LFSRs are ANDed together and then XORed with each other to produce the final output bits of the generator. The output signal *c*(*t*) is produced according to:

$$C(t) = \sum_{i=0}^{l-1} a(t+i).\ b(dt+i)$$

The speed factor *d* is variable and is used as a part of the key of secrecy.

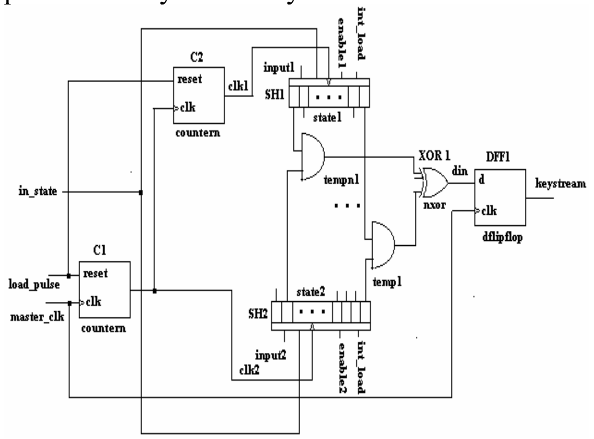


Figure 3.8 Schematic Diagram of the Massey Rueppel's Multispeed Generator

Massey Rueppel's Generator

S. No	Combination			Logic element utilization	
	N_1	n_2	D	No of cells used out of 1320	Percentage utilization
1.	6	8	8	98	7
2.	13	17	8	117	8
3.	14	18	8	128	9
4.	17	23	8	134	10
5.	18	24	8	139	10
6.	31	37	8	167	12
7.	32	38	8	171	12
8.	41	43	8	187	14
9.	42	44	8	190	14
10.	59	61	8	228	17
11.	60	62	8	228	17

Table 3.4 LC utilization of Massey Ruppel Generator

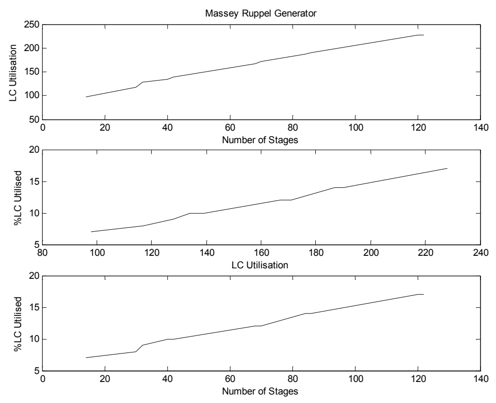


Figure 3.9 LC utilization of Massey Ruppel Generator

3.2.2.2 The Rueppel's Self Decimator[7]

A Self Decimator is a generator that controls its own clock. Rueppel proposed a self-decimator using a single LFSR. Figure 3.8 illustrates the arrangement. An LFSR with a primitive feedback polynomial is used to clock control itself in the following way:

When the output signal is 0, *d* clock pulses are applied to the LFSR; otherwise *k* clock pulses are applied. The effect is that some of the signals of the *m*-sequence generated by the LFSR are skipped or decimated in forming the output sequence, according to a pseudorandom rule determined by the m-sequence itself.

Parameters of the generator:

- The degree of the feedback polynomial of the LFSR is *n*.
- Clocking factors are *d* and *k* when the outputs are 0 and 1 respectively.

Optimal Conditions:

- The feedback polynomial of the LFSR is primitive.

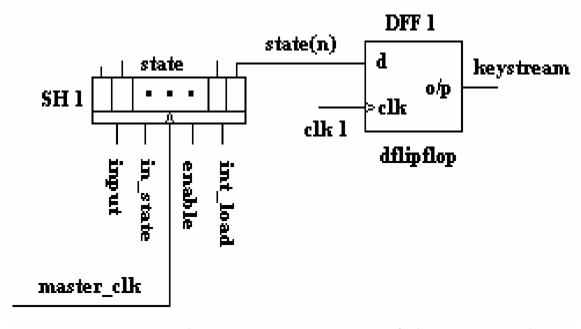


Figure 3.10 Schematic Diagram of the Rueppel's Self Decimator

Rueppel's Self Decimator

S.No.	Combination	Logic element utilization

	N	C_0	c_1	No of cells used out of 1320	Percentage utilization
1.	7	1	3	235	17
2.	8	1	3	238	18
3.	12	1	3	243	18
4.	13	1	3	244	18
5.	17	1	3	248	18
6.	18	1	3	251	19
7.	23	1	3	257	19
8.	37	1	3	276	20
9.	38	1	3	278	21
10.	43	1	3	284	21
11.	44	1	3	286	21
12.	61	1	3	306	23
13.	62	1	3	309	23

Table 3.5 LC utilization of Ruppel Self Decimator

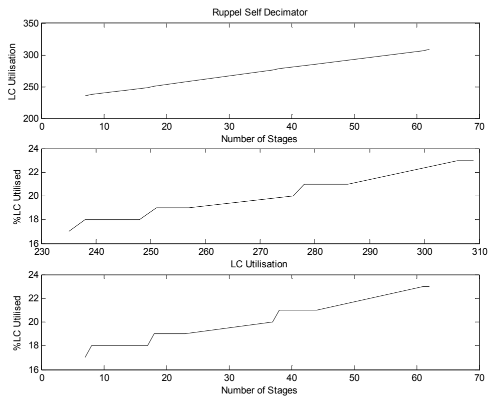


Figure 3.11 LC utilization of Ruppel Self Decimator

3.2.2.3 The Gollmann's Self Decimator[1]

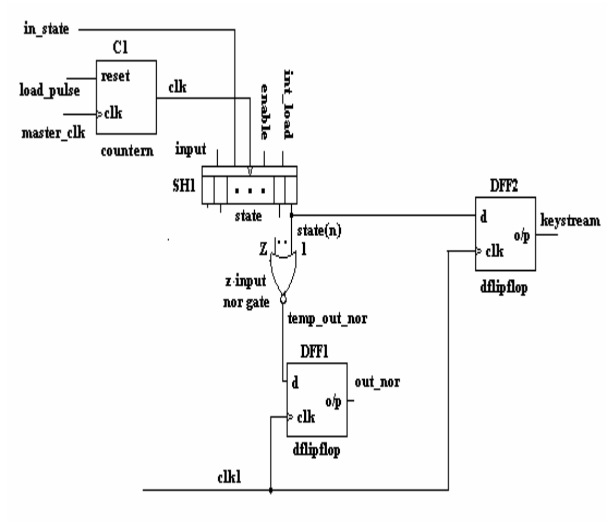
Chambers and Gollmann proposed the scheme illustrated in figure 3.9. By suitably choosing the number *n* of stages in the LFSR and the number *Z* of input ends to the transferring NOR function, the authors succeeded in producing a binary sequence prime period *p* and linear complexity equal to *p-1* or *p*. with the increase of the number *Z*, the resulting decimated sequence differs from the undisturbed *m*-sequence less and less.

Parameters of the generator:

- The degree of the feedback polynomial of the LFSR is *n*.
- *Z*: the number of inputs to the NOR gate.

Optimal Conditions:

- The feedback polynomial of the LFSR is primitve



3.12 Schematic Diagram of the Gollmann's Self Decimator

Gollman's Self Decimator

S. No	Combination		Logic element utilization	
	n	Z	No of cells used out of 1320	Percentage utilization
1.	11	5	264	20
2.	30	5	289	21
3.	41	5	302	22
4.	42	5	307	23
5.	47	5	308	23
6.	48	5	314	23
7.	67	5	336	25
8.	68	5	343	25

Table 3.6 LC utilization of Gollman's Self Decimator

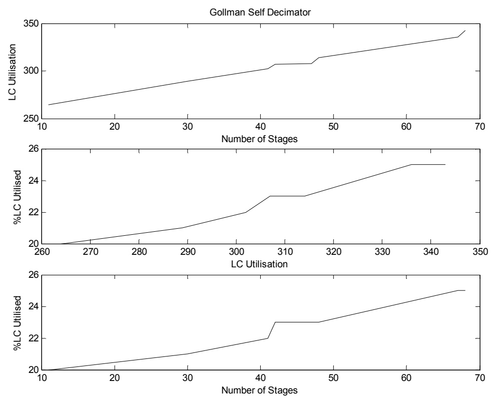


Figure 3.13 LC utilization of Gollman's Self Decimator

4 REALISATION ANALYSIS

In general stream ciphers using LFSRs utilizes the logic cells in proportion with the number of stages in the LFSRs. The cipher systems were then implemented on the FPGA using the tool as outlined in the previous section. The signals plain_in_enc, datain that was serially fed in data to load the *LFSRs*, load1 and load2 were supplied to the system via a test bench. The plain_out_dec signal that was the decrypted text emanating from the Decryptor block was compared with the plain_in_enc signal that was the plain text input to the encryptor block and the two signals were found to match in each cipher system. The waveforms obtained on the oscilloscope were found to emulate those in the simulation reports that were obtained on simulation prior to the FPGA implementation. The generators used in this work use some combinational logic to introduce nonlinear characteristics in the sequence generated by them. From the above graphs between the total Number of stages of LFSRs used in the Generator and LCs used in FPGA (figure 3.3(a), 3.5(a), 3.7(a), 3.9(a), 3.11(a), 3.13(a) , the graphs of (figure 3.3(b), 3.5(b), 3.7(b), 3.9(b), 3.11(b), 3.13(b)) between LCs used and %of the LCs used among the total LCs and Number of stages of LFSRs and from the graphs of (figure 3.3(c), 3.5(c), 3.7(c), 3.9(c), 3.11(c), 3.13(c)) %of the LCs used among the total LCs,)) is clear that these generators are having almost linearly increasing relationship between the number of stages in LFSRs, LC utilization and %of LC utilizated in FPGA. The perfect smoothness is not obtained due to the combinational logics present in the design.

5 FUTURE WORK AND FURHTER IMPROVEMENTS

The Stream ciphers design and implementation in hardware is an unexposed area. The realization of the stream cipher algorithms in a reconfigurable hardware is really a challenging work. In the implementation of the Stream ciphers, the challenge lies in the minimum hardware utilization, key stream generation at very fast rate (>8 Mbps), and synchronization with the distance end system. In this work the hardware design related to the resource utilization and the key stream generation speed are tested for a few generators. There is still lot more work related to the synchronization, the key transfer, the initialization of LFSR stages in the key stream generators have to be studied. Further work in these areas may lead to a design of fast secure key stream generator (>8 Mbps) in hardware for practical use.

Acknowledgement
Thanks are due to Sri Sanjay Burman Scientist –'F'CAIR, Rajbhavan Circle, Bangalore, for his valuable suggestions in design and implementation of this work. The authors also thank Director, CAIR for providing all the facilities to carry out this work.

References

1. Kencheng Zeng, Chung-Huang Yang, Dah-Yea-Wei and T.R.N. Rao, University of Southwestern Louisiana "*Pseudorandom Bit Generators In Stream-Cipher Cryptography*".
2. Fred Piper "Stream Ciphers," *Springer Verlag Lecture notes in Computer Science* 1998 pp. 181-188
3. Rainer A. Rueppel, "*Linear Complexity and Random Sequences*," *Springer Verlag Lecture notes in Computer Science* 1998 pp. 167-188
4. S.M. Jennings, "*Multiplexed Sequences: Some Properties of the Minimum Polynomial*" *Proc Workshop on Cryptography, No. 149*, *Springer Verlag Lecture notes in Computer Science,* New York 1982 pp. 189-206
5. T. Beth and F.C. Piper, "The *Stop-and-Go Generator*," *Proc Eurocrypt '84, Springer Verlag Lecture notes in Computer Science,* No. 209 New York 1982 pp. 88-92.
6. Dieter Gollman, "*Pseudorandom Properties of Cascade Connections of Clock Controlled Shift Registers*," *Springer Verlag Lecture notes in Computer Science* 1998 pp.93-98.
7. R.A. Rueppel, "*When Shift Registers Clock Themselves*," *Proc Eurocrypt '87, Springer Verlag Lecture notes in Computer Science,* No. 304, New York 1987, pp.53-64.
8. Menzez et.al. "*Hand book of Applied Cryptography*", CRC Press, 1997
9. B. Schneier. "Applied Cryptography". John Wiley & Sons Inc., New York, New York,USA, 2nd edition, 1996.