

## Overview

RC4 is a pseudo random sequence generation algorithm developed in 1987 by Ron Rivest for RSA Data Security Inc. The RC4 is a simple algorithm for random sequence generation. This algorithm is immune to differential and linear cryptanalysis. This design example illustrates the implementation of RC4 based PRNG using iCE65 FPGAs. RC4 based PRNG HDL code is a generic implementation which supports a variable key size of 5 to 10 bytes.

## Description

RC4 is a variable key-size stream cipher. RC4 generates a pseudorandom stream of bits which for encryption, is combined with the plaintext using bit-wise exclusive-or, decryption is performed the same way. RC4 stream cipher has two phases, the key set-up and the keystream generation, which are shown below:

The key used in RC4 simply permutes a 256 byte array. Once the permutation procedure is finished, the key is never used again. After that, a byte stream is selected from 256-byte array in a systematic fashion, and it is used to encrypt or decrypt data.

RC4 used a variable-length key, where key length is from 1 to 156 bytes, and a 256 byte array  $s$  is used in RC4. In initialization process  $S[i]$  is  $i$ , where  $i$  is from 0 to 255, then we put the variable-length key in a array  $K$ . The same key is used to do initial permutation of  $S$ .

```
// Initialization
for i = 0 to N
  S[i] = i;
j = 0;
// key shuffling
for i = 0 to N
  j = (j + S[i] + K[i mod l]) mod N; // K is secret Key and l is K size in bytes
  Swap(S[i],S[j]);
```

Finally, we choose 1-byte value from the permuted  $S$ , and at the same time swap some two bytes in the  $S$ .

```
//Stream generation
i = j = 0
i = (i + 1) mod 256
j = (j + S[i]) mod 256
swap S[i] and S[j]
t = (S[i] + S[j]) mod 256
PRNG = S[t] // which is the output PRNG
```

Figure 1 shows the basic block diagram of RC4 based PRNG implementation.

Figure 1: RC4 based PRNG generator

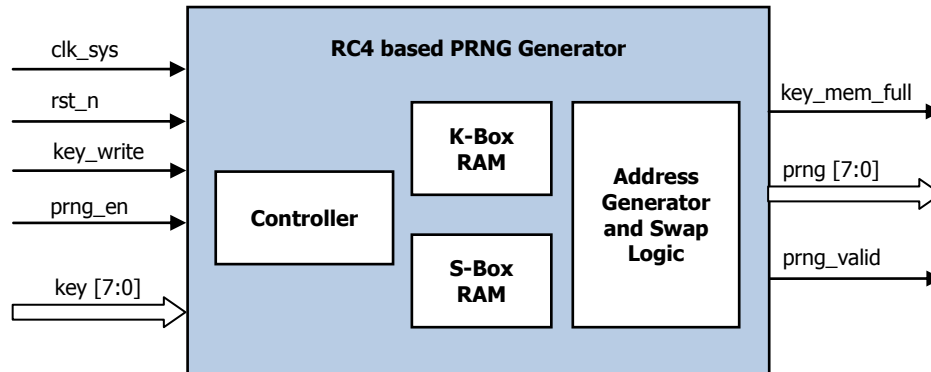


Table 1: Pin Description

Pin	Direction	IO Bank	Voltage(V)	Description
clk_sys	Input	0/1/2/3*	1.8/2.5/3.3	System clock
rst_n	Input	0/1/2/3*	1.8/2.5/3.3	System Reset, Active low
key_write	Input	0/1/2/3*	1.8/2.5/3.3	Key write enable, Active low
prng_en	Input	0/1/2/3*	1.8/2.5/3.3	PRNG generation, Active low
Key[7:0]	Input	0/1/2/3*	1.8/2.5/3.3	Key Input
key_mem_full	Output	0/1/2/3*	1.8/2.5/3.3	Key memory full indicator, Active High
prng[7:0]	Output	0/1/2/3*	1.8/2.5/3.3	PRNG output
prng_valid	Output	0/1/2/3*	1.8/2.5/3.3	PRNG output valid, Active High

\* Note that Bank 3 does not support 3.3V I/O.

## Implementation

Control FSM controls the process of key Set up phase and Key Stream Generation Phase

### Key Setup phase

A low on key\_write initiates key set up phase. When key\_write goes low K-Box RAM starts storing the key, key\_mem\_full output goes high after writing key of variable length into K-Box RAM. S-Box RAM is initialized to 0 to 255 linearly. After the initialization of S-Box RAM and K-Box RAM shuffling of data is done for 255 iterations, and a new address is computed using the key; swapping of data takes place in S-Box RAM.

### Key Stream Generation Phase

Key stream Generation phase gets initiated after the completion of key set up phase if prng\_en is low and key\_write is high. This phase also generates new address but key is not used in this phase, data shuffling is done just like key set up Phase. prng [7:0] output is obtained with high on prng\_valid output. If prng\_en is made high and key\_write as low, indicating the presence of new key, Key Set up Phase gets initialized for this new key.

Figure 2: Sample waveform of key writing

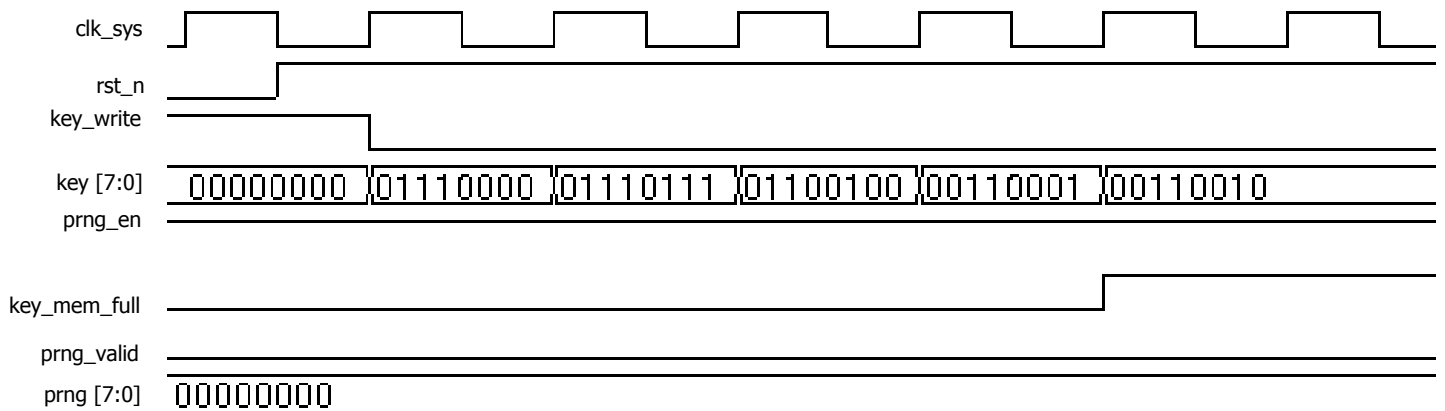


Figure 3: Sample waveform of valid prng output

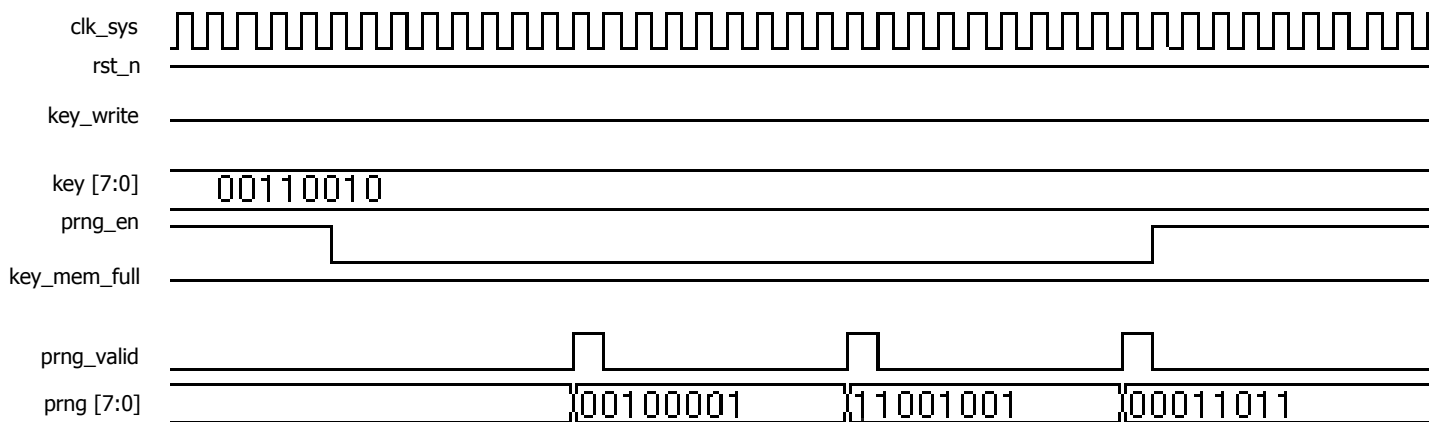


Table 2: Resource Utilization

Device	Flip-Flops	LUTs	RAMs	IO Cells
<b>iCE65L04-UCB284</b>	151	279	2	22

Table 2 shows the resource utilization summary of this design when implemented on an iCE65 FPGA. This data was generated with default implementation options.

### Conclusion

This design example demonstrates the implementation of RC4 based PRNG on iCE FPGAs. Low power characteristic of iCE FPGA makes this design example a best choice for encryption and decryption applications.

### Revision History

Version	Date	Description
<b>1.2</b>	23-DEC-2008	Formatting update, disclaimers, address change, design name change.
<b>1.1</b>	28-NOV-2008	Initial draft.

## Disclaimer

SILICONBLUE TECHNOLOGIES PROVIDES THIS APPLICATION NOTE TO YOU "AS-IS". ALL WARRANTIES, REPRESENTATIONS, OR GUARANTEES OF ANY KIND (WHETHER EXPRESS, IMPLIED, OR STATUTORY) INCLUDING, WITHOUT LIMITATION, WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR A PARTICULAR PURPOSE, ARE SPECIFICALLY DISCLAIMED.

LIMITATION OF LIABILITY: SUBJECT TO APPLICABLE LAWS: (1) IN NO EVENT WILL SILICONBLUE TECHNOLOGIES OR ITS LICENSORS BE LIABLE FOR ANY LOSS OF DATA, LOST PROFITS, COST OF PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES, OR FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL OR INDIRECT DAMAGES ARISING FROM THE USE OR IMPLEMENTATION OF THE APPLICATION NOTE, IN WHOLE OR IN PART, HOWEVER CAUSED AND UNDER ANY THEORY OF LIABILITY; (2) THIS LIMITATION WILL APPLY EVEN IF SILICONBLUE TECHNOLOGIES HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES; (3) THIS LIMITATION SHALL APPLY NOTWITHSTANDING THE FAILURE OF THE ESSENTIAL PURPOSE OF ANY LIMITED REMEDIES HEREIN.



SiliconBlue Technologies Corporation

3255-7 Scott Blvd., Suite 101  
Santa Clara, CA 95054

Tel: 408-727-6101

Fax: 408-727-6085

[www.SiliconBlueTech.com](http://www.SiliconBlueTech.com)