

Overview

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

Features

- RC4 variable key-size stream-cipher algorithm
- Configurable keys of size 5 to 10 bytes
- Delivers pseudo random numbers as byte streams
- Hardware tested for encryption and decryption
- Useful in stream cipher crypto engines.
- IP-XACT version 1.2 compliant

Resource Utilization

Table 1: Resource Utilization

LUTs	Registers	Memory	I/Os	GBs
206	111	2	0	0

Note: Resource Utilization is based on iCEcube2 2010.12.14671

System Block Diagram

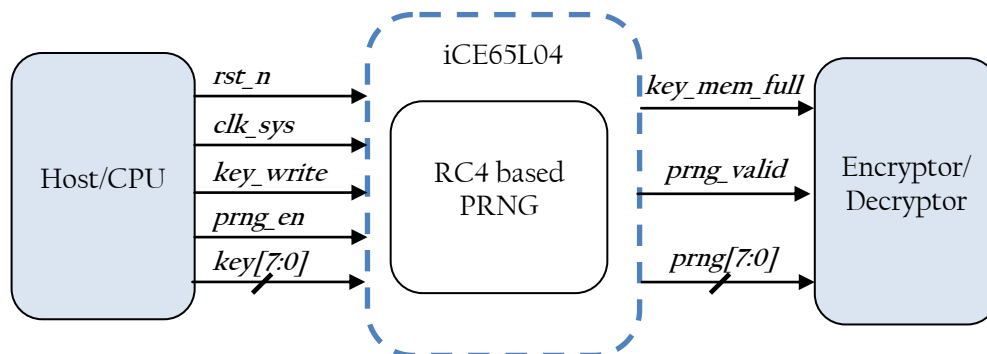


Figure 1: System Block Diagram

Functional Block Diagram

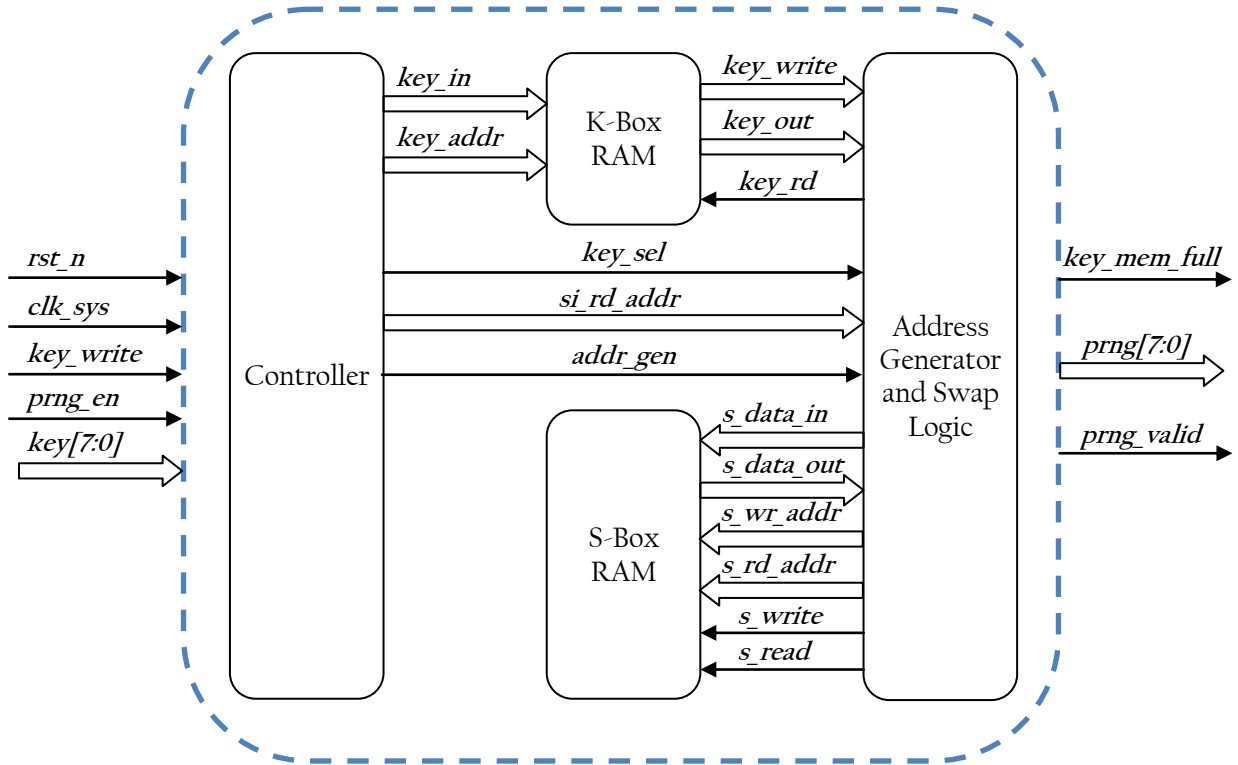


Figure 2: Functional Block Diagram

Design Interface

Table 1: Pin Description

Signal Name	Pin Type	Signal Description
clk_sys	Input	System clock, at 32 MHz.
rst_n	Input	Active low asynchronous reset.
key_write	Input	Active low key write enable
prng_en	Input	PRNG generation enable signal, generates PRNG byte streams when held low.
key[7:0]	Input	Key input, Multi-byte key write supported with a low key_write for multiple cycles.
key_mem_full	Output	Key Memory full indicator A high on this signal indicates that the key memory is full.
prng[7:0]	Output	PRNG output
prng_valid	Output	PRNG output valid A high on this signal indicates a valid PRNG output.

Configurable Parameters

- KEY_LENGTH – This parameter configures the length of the key to be used. Its supports values of range 5 to 10
Default value is 8

Register Map

This design does not have any user accessible registers or memory.

Design Details

RC4 is a variable key-size stream cipher. RC4 generates a pseudo-random stream of bits which for encryption, is combined with the plain text using bit-wise exclusive-or, decryption is performed in the same way. RC4 stream cipher has two phases, the key set up and the key stream generation.

The key 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 uses 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 an 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
```

$i = (i + 1) \bmod 256$

$j = (j + S[i]) \bmod 256$

swap $S[i]$ and $S[j]$

$t = (S[i] + S[j]) \bmod 256$

PRNG = $S[t]$ // which is the output PRNG

Implementation

Control FSM controls the process of key set up phase and key stream generation phase

Key Set up 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 256 iterations, where new address is computed for each iteration using the key and 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` is low, indicating the presence of new key, key set up phase gets initialized for this new key.

Initialization Conditions

No user specific initialization conditions, except that the `rst_n` must be held low initially to bring-up the design in a correct operating state.

Timing Diagram

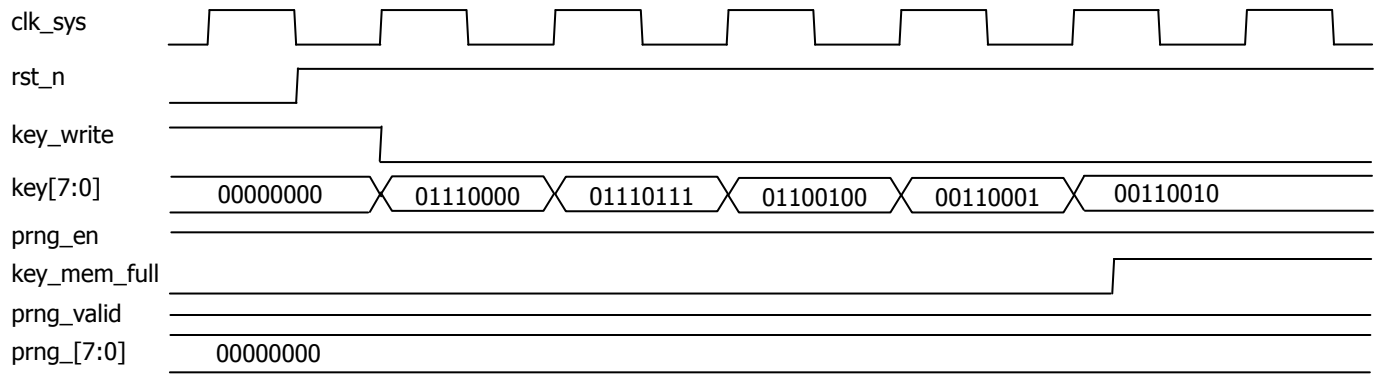


Figure 3: Sample waveform of key writing

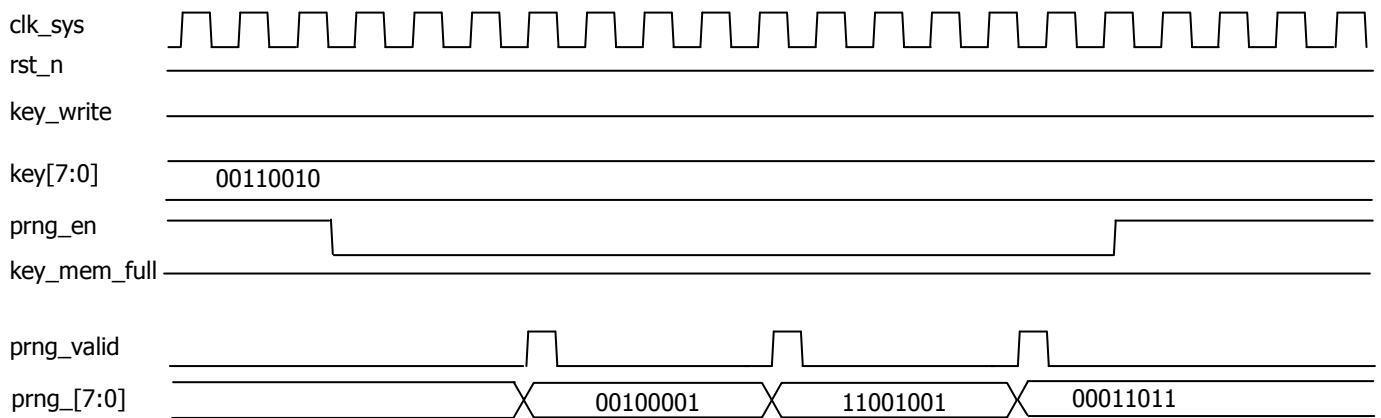
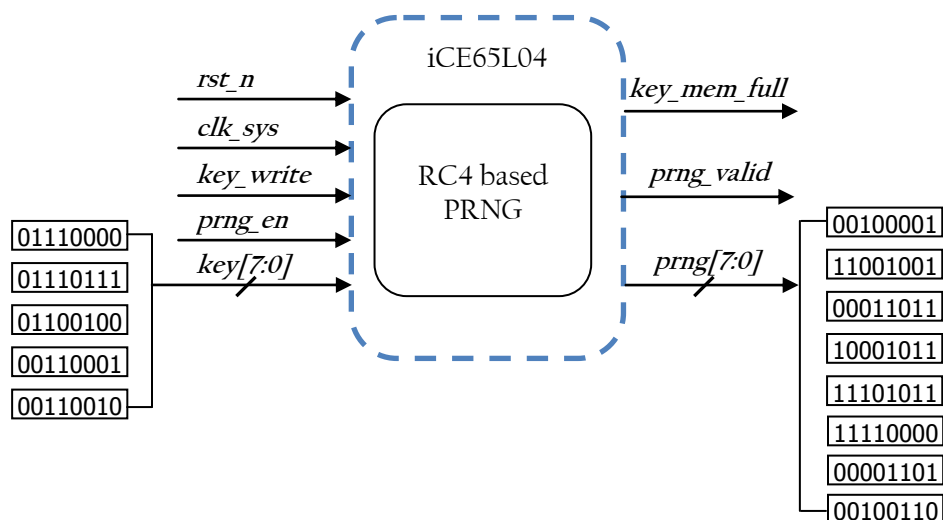


Figure 4: Sample waveform of valid prng output

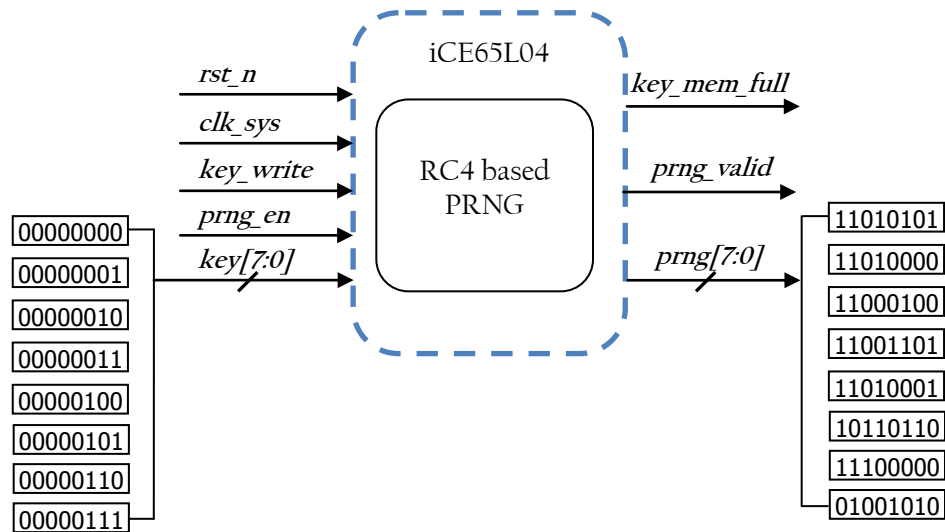
Usage Examples

Example usages are shown below:

Example #1



Example #2



System Designer Flow

RC4 Based PRNG is compatible with System Designer/IP-XACT 1.2. Following parameters can be configured in the System Designer environment:

- **KEY_LENGTH** – This parameter configures the length of the key to be used. It supports values of range 5 to 10. Default value is 8.

The System Designer flow is as follows,

1. Launch the System Designer from Synplify Pro using menu 'Import -> Launch System Designer'.
2. Create a new project (open an existing old project, as necessary) and import the IP-XACT XML file.
3. Drag and place the component from the 'Library' pane to the 'Design' pane.
4. To change the key length, right-click on the component instance, and click on "Open Configuration". Go to "Edit Instance Parameters" tab, change the "KEY_LENGTH" parameter. Click on the "Apply" button, and then close it.
5. Click on the "Generate Files" button, which generates the necessary files required for synthesis and simulation.
6. Go to Synplify Pro and click on the "Run" button to synthesize the System Designer generated files. Synplify Pro generates all the necessary files for P&R in iCECube.

References

The following references were used in the creation of this design:

- SiliconBlue Technologies, Inc. “[iCE65 Ultra Low-Power mobileFPGA Family](#)” datasheet (26-MAY-2010).
- Wikipedia : <http://en.wikipedia.org/wiki/RC4>

Revision History

Version	Date	Description
1.0	09-SEP-2010	Initial Draft Document
1.1	08-DEC-2010	IP-XACT format Update

Disclaimer

Copyright © 2007–2009 by SiliconBlue Technologies LTD. All rights reserved. SiliconBlue is a registered trademark of SiliconBlue Technologies LTD in the United States. Specific device designations, and all other words and logos that are identified as trademarks are, unless noted otherwise, the trademarks of SiliconBlue Technologies LTD. All other product or service names are the property of their respective holders. SiliconBlue products are protected under numerous United States and foreign patents and pending applications, maskwork rights, and copyrights. SiliconBlue warrants performance of its semiconductor products to current specifications in accordance with SiliconBlue's standard warranty, but reserves the right to make changes to any products and services at any time without notice. SiliconBlue assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by SiliconBlue Technologies LTD. SiliconBlue customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.



SiliconBlue Technologies Corporation

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

Tel: 408-727-6101
Fax: 408-727-6085

www.SiliconBlueTech.com