SiLVUS TECHNOLOGIES

SP 800-90B Non-Proprietary Public Use Document

Document Version: v1.1

Date: March 15, 2023

Entropy Source: Silvus Clock Jitter Entropy Module

Prepared for:

Silvus Technologies
10990 Wilshire Blvd #1175
Los Angeles, CA 90024

Prepared by:

KeyPair Consulting Inc.
987 Osos Street
San Luis Obispo, CA 93401
United States

Change History

Version	Change Description	Date	Author
0.1	Template.	August 3, 2022	KeyPair
0.2	First pass at required detail.	August 9, 2022	Silvus
0.3	Updated to move boilerplate text to comments, basic physical security statement, minor aesthetic details.	August 11, 2022	KeyPair
0.4	Part and version information updates.	October 25, 2022	KeyPair
0.5	Incorporated EAR excerpts.	October 27, 2022	KeyPair
0.6	Initial Silvus details.	November 14, 2022	Silvus
0.7	Merged Silvus details, updated Required Testing.	November 15, 2022	KeyPair
1.0	Finalized version	January 19, 2023	Silvus, UL
1.1	Page numbers corrected in Table of Contents; unnecessary information removed from Section 6 and Table 4 as a result of CMVP Comments.	March 14, 2023	Silvus, UL

References

Ref	Title	Date	Author
[90B]	NIST SP 800-90B, Recommendation for the Entropy Sources Used for Random Bit Generation	10-Jan-2018	NIST CT
[140IG]	Implementation Guidance for FIPS 140-3 and the Cryptographic Module Validation Program	21-Sept-2020	CMVP
[ESVMM]	Entropy Source Validation, program Management Manual (under development)	TBD	ESV
[JEDSGN]	CPU Time Jitter Based Non-Physical True Random Number Generator	July 1, 2022	S.Müller Chronox.de
[Hill 2020]	SP 800-90B Refinements: Validation Process, Estimator Confidence Intervals, and Assessment Stability. ICMC	2020	Joshua E. Hill
[Müller 2022]	CPU Time Jitter Based Non-Physical True Random Number Generator	July 1, 2022	Stephan Müller
[JEnt-MemOnly]	Jitterentropy library with MemOnly updates. https://github.com/joshuaehill/jitterentropy-library/tree/MemOnly		

Table of Contents

1 Description

Silvus Technologies, Inc. (Silvus) is using the Jitter Entropy Library MemOnly branch, hereafter referred to as "JEnt-MemOnly", as a non-physical entropy source to generate entropy input for instantiation and reseed of an SP 800-90A compliant DRBG in the Silvus Clock Jitter Entropy Module (SCJEM). JEnt-MemOnly is based on the jitterentropy-library v3.4.1, which is documented in [Müller 2022, Section 3]. The JEnt-MemOnly design is depicted in Figure 1.

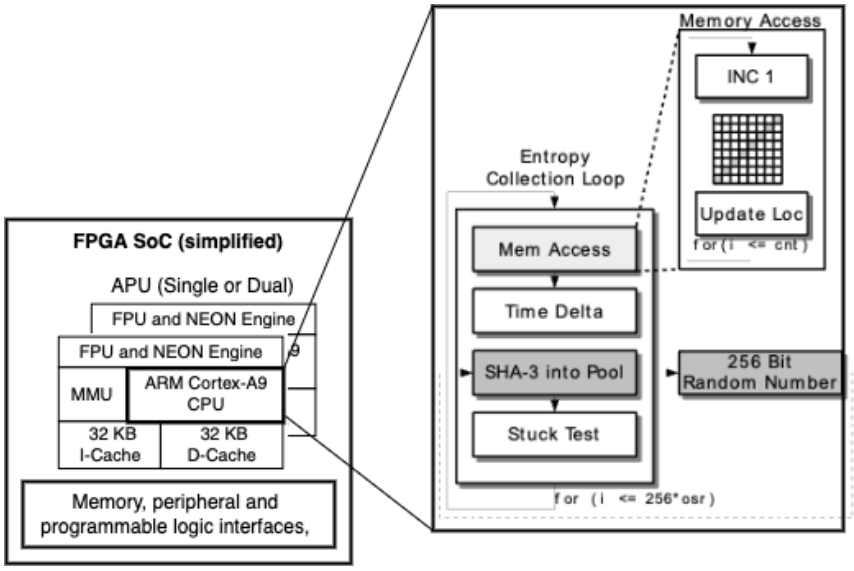

Figure 1: Entropy Source (Adapted from [Müller 2022, Figure 3.1])

The JEnt-MemOnly design includes elements that map to the conceptual components contained within an SP 800-90B entropy source:

- The Memory Access (MemAccess) primary noise source (memory timing)
- An additional noise source (Overall Timing)
- Health tests (depicted as the "Stuck Test" block in Figure 1)
- A conditioning algorithm (SHA-3)

Table 1: Evaluated Entropy Source Specification

Identifier	Details			
Entropy Source Name	SCJEM	SCJEM	SCJEM	SCJEM
Part Number	SC4240EP-235467-BB	SC4480E-235467-SBST	SL4210-235-SB	SM4210-235-SB
Hardware Revision	B7	B1	C3	B2
Firmware Version	4.1.0.0 4	4.1.0.0	4.1.0.0	4.1.0.0
Entropy Category	Non-physical (NP)	Non-physical (NP)	Non-physical (NP)	Non-physical (NP)
Test Platform(s)	Xilinx FPGA fabric, dual core CortexA9 CPU running Linux kernel version 3.17.	Xilinx FPGA fabric, dual core CortexA9 CPU running Linux kernel version 3.17.	Xilinx FPGA fabric, dual core CortexA9 CPU running Linux kernel version 3.17.	Xilinx FPGA fabric, dual core CortexA9 CPU running Linux kernel version 3.17.
Entropy Estimation Track (per SP 800-90B §3.1.2)	Non-IID	Non-IID	Non-IID	Non-IID

2 Security Boundary

The Silvus Clock Jitter Entropy Module entropy source in the context of the FPGA SoC Application Processor Unit (APU, and ARM Cortex-A9) is depicted in Figure 1.

3 Operating Conditions

The Entropy-relevant operating conditions for all entropy source variants listed in Table 1 are given in Table 2.

Table 2: Entropy-Relevant Operating Conditions

Parameter	Value	Description
Temperature	-40C to 65C	Operating temperature range.
Voltage	9 to 20 VDC	Operating voltage range.
Clock speed	400 Mhz	CPU clock speed.

4 Configuration Settings

Table 3. Entropy-Relevant Parameters

Parameter	Meaning	Type	Value
JENT_CONF_ENABLE_INTERNAL_TIMER	Allow the pthread-based timer.	Macro	Not Defined
JENT_HEALTH_LAG_PREDICTOR	Use the Lag-predictor-based health test.	Macro	Defined
JENT_LAG_WINDOW_SIZE	Size of the Lag-predictor-based health test window.	Macro	131072
JENT_LAG_HISTORY_SIZE	Size of the Lag-predictor-based history.	Macro	8
JENT_DISTRIBUTION_MIN	Lower bound for the selected memory sub-distribution.	Macro	1392
JENT_DISTRIBUTION_MAX	Upper bound for the selected memory sub-distribution.	Macro	1647
JENT_MEMORY_SIZE_EXP	Exponent of selected memory size (i.e., 2^{22} bytes will be used).	Macro	22
JENT_MEMORY_DEPTH_EXP	Exponent of selected memory depth (i.e., there is no decimation).	Macro	0
JENT_APT_WINDOW_SIZE	APT Window size.	Macro	512
JENT_HASHLOOP_EXP	Exponent of number of hash loops (i.e., there is 1 hash loop per output).	Macro	0
JENT_MEMACCESSLOOP_EXP	Exponent of number of MemAccess loops (i.e., there is 1 MemAccess loop per output).	Macro	0
JENT_DIST_WINDOW	Size of the window for the Distribution Health Test.	Macro	10000
JENT_POWERUP_TESTLOOPCOUNT	A lower bound for the number of values to test on initialization (JENT_POWERUP_TESTLOOPCOUNT + CLEARCACHE are performed).	Macro	1024
CLEARCACHE	An additional number of values to perform to set branch prediction and caches.	Macro	100
ENTROPY_SAFETY_FACTOR	The number of extra bits of min entropy required to make a "full entropy" claim. Used only in FIPS mode.	Macro	64
osr	Oversample Rate. The code presumes that the lower bound for the min entropy is 1/osr.	Variable	1
fips_enabled	Flag that controls error mode reporting and use of the ENTROPY_SAFETY_FACTOR.	Variable	1
enable_notime	Use the constructed pthread timer.	Variable	0
Optimizer Setting	The compiler optimizer setting has a substantial impact on the resulting distribution.[1]	Compiler Flag	-O0

[1] The impact of the optimizer on this library is extensively discussed on the library's GitHub repository Issue #21.

5 Physical Security Mechanisms

The Silvus Clock Jitter Entropy Module entropy source operates within the physical protections of the associate FPGA package, a commercial plastic ball grid array package. The modules that incorporate the SCJEM entropy source incorporate physical security protections:

- Production-grade components and production-grade opaque enclosure.
- Tamper-evident seals applied during manufacturing.

6 Conceptual Interfaces

The entropy source is depicted in Figure 1. The entire design of this entropy source is documented in [Müller 2022, Section 3.1] and [JEnt-MemOnly]. The changes present in [JEnt-MemOnly] were summarized in [Hill 2022]. At a high level, the entropy source is initialized by running the `jent_entropy_init` function. This function in turn calls `jent_time_entropy_init` which is responsible for determining if the counter is sufficiently fine-grained to support the library, and which performs the required power-on health tests.

After the library has been initialized, new instances can be allocated using the `jent_entropy_collector_alloc` function. This function allocates the memory necessary for a JEnt instance, establishes the value of osr for this instance, and allows the user to request various behaviors by passing in the flags listed in Table 4.

Table 4. Supported JEnt Flags

Flag	Description
JENT_DISABLE_INTERNAL_TIMER	Forces the use of the hardware counter.
JENT_FORCE_FIPS	Forces the fips_enabled flag.

The jitterentropy library source code is compiled in directly into fips_entropy_gcc.so, an OpenSSL seed provider. It includes the function `setup_drbg` which every crypto module invokes to switch to FIPS mode.

The JEnt instance can then be used to request entropy using the `jent_read_entropy` function. This function repeatedly produces blocks of 256 bits of conditioned data using the `jent_random_data` function until at least the amount of requested data has been produced.

The `jent_random_data` function first makes an initialization call to `jent_measure_jitter` to establish an initial value for the time, and then iteratively calls `jent_measure_jitter` in a loop until (256+ ENTROPY_SAFETY_FACTOR)*osr non-stuck delta values have been input into the conditioner.

7 Min-Entropy Rate

In accordance with the Entropy Assessment Report, this entropy source has full-entropy output; that is, the analysis supports the claim of 256 bits of min entropy per 256-bit output block. These blocks are the output of a vetted conditioning function and can be subdivided further (as per SP 800-90B). If these blocks are subdivided, then every byte from the block can be treated as possessing at least 8 bits of min entropy, and the min entropy of any truncated sub-portion of the 256-bit output block is linearly scaled with the length of the retained sub-portion.

This entropy source is used only to seed an SP 800-90A compliant DRBG, providing 512 bits of entropy input, in excess of the requirement for entropy input and nonce (384 bits).

8 Health Tests

All health tests are essentially continuous health tests, and are tested within the `jent_stuck` function. This function performs a modified version of the Repetition Count Test (RCT), an Adaptative Proportion Test (APT), a Lag Health Test, and a Distribution Health Test.

The Lag Health Test is performed by attempting to predict the current symbol using the prior 8 symbols. If the most successful lag (delay) becomes too successful in predicting the current output, or if the test is globally more successful than expected, then the Lag Health Test fails.

The Distribution Health Test operates on data prior to the selection of a sub-distribution or any decimation. If the proportion of pre-raw data is too low, then the Distribution Health Test fails.

When the `fips_mode` flag is set, calls to the `jent_health_failure` function return with the current `health_failure` flag state. If fips_mode is not set, then this function always indicates that no failure has occurred. The `health_failure` flag indicates a persistent error for the JEnt instance used, and this flag cannot be reset. For an instance in FIPS mode, it is only possible to continue using the library for entropy production if a new JEnt instance is created.

The targeted cutoff parameters for the APT, RCT and Lag Predictor Health Test are dependent on the setting of osr. In the FIPS mode, only data that passes all health tests can be integrated into the conditioner.

The start-up tests run on 1124 consecutive samples using a subset of the continuous health tests, namely the APT, RCT, and Lag Predictor Health Tests. The samples used by the start-up health test are discarded.

On-demand testing is performed by allocating a new JEnt handle, which triggers the start-up tests. The samples used by the on-demand health test (effectively the start-up health test) are discarded.

There is no mechanism to clear an error state short of re-instantiating a new entropy source.

9 Maintenance

The Silvus Clock Jitter Entropy Module entropy source does not require maintenance.

10 Required Testing

The JEnt-MemOnly entropy source was tested in accordance with SP 800-90B requirements. The data collection was performed by Silvus in advance of testing from an instance of the entropy source. Raw and restart data was collected using the jitterentropy-hashtime utility, compiled consistent with the Table 3 configuration settings. Restart data was collected following device reboot.

No further testing is required.