Chapter 14
Digital Random Number Generator Hardware Accelerator IP-Core for Security Applications

Luca Baldanzi, Luca Crocetti, Francesco Falaschi, Jacopo Belli, Luca Fanucci and Sergio Saponara

Abstract Random numbers are widely employed in cryptography and security applications, and they represent one of the main aspects to take care of along a security chain. They are employed for creation of encryption keys, and if generation process is weak, the whole chain can be compromised: weaknesses could be exploited to retrieve the key, thus breaking even the strongest cipher. This paper presents the architecture of a digital Random Number Generator (RNG) IP-core to be employed as hardware accelerator for cryptographically secure applications. Such design has been developed starting from specifications based on literature and standards, and in order to assess the randomness degree of generated output, it has been successfully validated through the official NIST Statistical Test Suite. Finally the RNG IP-core has been characterized on Field Programmable Gate Array (FPGA) and ASIC standard-cell technologies: on Intel Stratix IV FPGA it offers a throughput of 720 Mbps requiring up to 6000 Adaptive Logic Modules, while on 45 nm it reaches a throughput of 4 Gbps with a complexity of 119 kGE.

L. Baldanzi · L. Crocetti · F. Falaschi (✉) · J. Belli · L. Fanucci · S. Saponara
Department of Information Engineering, University of Pisa, Via G. Caruso, 16, 56122 Pisa, Italy
e-mail: francesco.falaschi@phd.unipi.it

L. Baldanzi
e-mail: luca.baldanzi@ing.unipi.it

L. Crocetti
e-mail: luca.crocetti@phd.unipi.it

J. Belli
e-mail: jacopo.belli23@gmail.com

L. Fanucci
e-mail: luca.fanucci@unipi.it

S. Saponara
e-mail: sergio.saponara@unipi.it

S. Saponara and A. De Gloria (eds.), *Applications in Electronics Pervading Industry, Environment and Society*, Lecture Notes in Electrical Engineering 627,
https://doi.org/10.1007/978-3-030-37277-4_14

14.1 Introduction

In modern cryptography one of the fundamental primitives to be employed is the Random Number Generator (RNG), the component in charge of generation of arbitrary length random bit sequences. It represents the core part for several security applications which are required to ensure authentication, confidentiality and message integrity for a broad range of activities, such as payments, on-line authentication, instant messaging and operating systems updates [4]. The creation of cryptographic keys requires a high degree of randomness so that an attacker is unable to derive the secret key of a cipher thus compromising the whole chain, authentication protocols nonces represent a valid countermeasure against replay attacks, in digital signature random numbers prevent attackers to derive private keys [3].

During the last decades, several circuits have been proposed to cope with generation of RNG sequence, in particular the True Random Number (or Bit) Generators (TRNGs) which are based on analog noise as physical source to generate random bits [1–7]. Such devices have a high-quality output, but they are affected by significant drawbacks, because they typically offer low throughput or require high power consumption. Moreover, they can be unreliable for long term use due to unexpected behaviors caused by changes in the device operating conditions. These are strong limitations especially considering the target to be employed in high performances and high complexity digital integrated systems such as hardware accelerators.

The limitations of TRNG devices can be worked around by implementing RNGs as Deterministic Random Bit Generators (DRBGs): in this case the output sequences are generated by means of deterministic algorithms instead of random processes, therefore in order to guarantee the expected level of randomness it is required to periodically give a new seed to such DRBG mechanisms (i.e., *reseed* operation, high entropy content is given to the deterministic algorithm to restart the sequence generation). This allow to pursue the requirement of indistinguishability between the output bit sequence and truly random sequence.

The reminder of this paper is organized as it follows: Sect. 14.2 presents the trade-off analysis among the different algorithms suitable for DRBG module, Sect. 14.3 describes the DRBG design architecture, Sect. 14.4 collects the characterization results, and Sect. 14.5 discusses about conclusions of this work.

14.2 DRBG Algorithms Trade-Off Analysis

As already mentioned, NIST has approved a certain number of DRBG mechanisms [2]: those mechanisms are based on Hash functions (SHA, Secure Hash Algorithm), keyed-Hash Message Authentication Code (HMAC), and Counter (CTR) mode of Advanced Encryption Standard (AES) and Triple Data Encryption Standard (TDES), and they are briefly presented, focusing on performance evaluation in terms of security strength and hardware implementation.

Hash DRBG family is based on SHA1 and SHA2 functions, but only SHA2 cryptographic primitives are taken into exam since SHA1 offers low security strength and it is considered outdated. The parameters related to a DRBG mechanism based on SHA2 Hash function are reported in Table 14.1.

CTR[1] DRBG mechanism is based onto a block cipher core used in *counter mode*. The parameters of this mechanism are listed in Table 14.2.

Concerning Hash DRBG, the characteristics of available SHA2 IP core are listed in Table 14.3. SHA-224 and SHA-384 are discarded from the options, since they offer a shorter output block keeping area and latency equal to respectively SHA-256 and SHA-512. The two remaining functions show some differences:

- SHA-256 has lower latency per block than SHA-512 but the latter offers a higher throughput since it provides 512 bit every 80 clock cycles;

Table 14.1 Hash DRBG mechanisms parameters (SHA2 only) [2]

	SHA algorithm			
	SHA-224	SHA-256	SHA-384	SHA-512
Highest security strength	192	256	256	256
Output block length (*outlen*) (bits)	224	256	384	512
Min. entropy for *Instance* and *Reseed* (bits)	192	256	256	256
Seed length (*seedlen*) bits	440	440	888	888
Max. num. of bit per request	2^{19}	2^{19}	2^{19}	2^{19}
Max. num. of requests between *Reseeds*	2^{48}	2^{48}	2^{48}	2^{48}

Table 14.2 CTR DRBG mechanisms parameters

	AES Algorithm			
	3Key TDEA	AES-128	AES-192	AES-256
Highest security strength	112	128	192	256
Input/output block length (*blocklen*) (bits)	64	128	128	128
Key length (*keylen*)	168	128	192	256
Counter field length (*ctr_len*)	$4 \leq ctr\_len \leq blocklen$			
Min. entropy for *Instance* and *Reseed* (bits)	112	128	192	256
Seed length (*seedlen*) (bits)	232	256	320	384
Max. num. of bit per request	min(B, 2^{13})	min(B, 2^{19})	min(B, 2^{19})	min(B, 2^{19})
Max. num. of requests between *Reseeds*	2^{48}	2^{48}	2^{48}	2^{48}

$B = (2^{ctrl\_len} - 4)\ blocklen$ [2]

[1]CTR is an abbreviation for *Counter*.

Table 14.3 SHA2 IP core specifications

SHA2 algorithm	Area (kGE)	Latency per block (clock cycles)	Output block size (bits)
SHA-224	15	64	224
SHA-256	15	64	256
SHA-384	30	80	384
SHA-512	30	80	512

Table 14.4 AES IP core specifications

AES algorithm	Area (kGE)	Latency per block (clock cycles)	Output block size (bits)
AES-128	11	11	128
AES-256	12.5	15	128

- comparing the areas, SHA-256 results to be more compact and this reflects also on internal state registers area footprint: as it can be seen in Table 14.1, the variable *seedlen* is 440 for SHA-256 and 888 for SHA-512; this implies that the internal state requires around 900 registers for the former and 1800 for the latter.

Now, the expected throughput of these two hash functions during generation phase in a Hash DRBG implementation can be calculated:

$$T_{SHA-256} = 256/64 \cdot f_{clk} \cdot n_{parallel\_core} = 4 \cdot f_{clk} \cdot n_{parallel\_core}\,\text{bit/s} \quad (1)$$

$$T_{SHA-512} = 512/80 \cdot f_{clk} \cdot n_{parallel\_core} = 6.4 \cdot f_{clk} \cdot n_{parallel\_core}\,\text{bit/s} \quad (2)$$

CTR DRBG proved to be best in class for both area and throughput. The characteristics of available AES IP core are presented in Table 14.4 for AES-128 and AES-256.

Since our focus is on highest level security strength implementations, only AES-256 is to be considered for the trade-off. As shown in the table, area is lower than SHA-256 and throughput is higher than SHA-512:

$$T_{AES-256} = 128/15 \cdot f_{clk} \cdot n_{parallel\_core} = 8.53 \cdot f_{clk} \cdot n_{parallel\_core}\,\text{bit/s} \quad (3)$$

Despite all these considerations, CTR DRBG has not been chosen to be implemented. The reason lays in the doubts about the effective capability of this mechanism to reach maximum security strength. In [8], the author claims that, while Hash-based DBRGs satisfy security requirements, block cipher-based ones should be avoided since the pseudo-random permutation inside each AES round coupled with the counter mode outputs a sequence which is indeed distinguishable from a random source. The choice ultimately fell on Hash DRBG, implemented with SHA-256

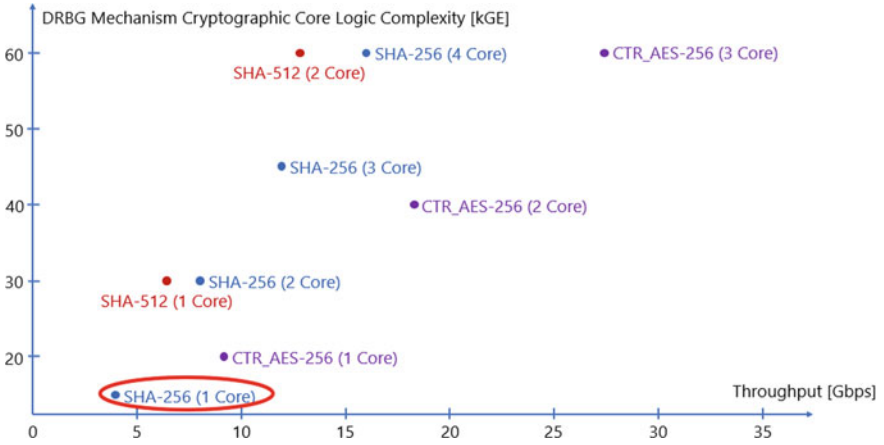

Fig. 14.1 Comparison between NIST approved DRBG mechanisms based on logic complexity in kGE and throughput

core. This ensures a compact implementation for the mechanism and the possibility to extend the design for supporting multiple cores to increase the throughput. Figure 14.1 reports the characteristics in terms of logic complexity and throughput of several DRBG implementations, relying on the available IP cores (SHA and AES) as primitives, their features when synthesizing on 45 nm standard-cell technology [9] and methods to construct DRBG using such primitives [2].

14.3 Hash DRBG Design Architecture

The design architecture of Hash DRBG with SHA-256 core is shown in Fig. 14.2, and it makes use of the following blocks:

- state registers for *V*, *C* and *Reseed counter*, with length respectively of 440, 440 and 20 bits, a 128-bit register to store an optional personalization string, for internal state randomization, and a 512-bit entropy register to store the input entropy content;
- a SHA-256 core with 512-bit input and 256-bit output, with a latency of 64 clock cycles;
- a serial adder with 440-bit inputs and modulo 440-bit output, which works in parallel with the SHA-256 core and stores the result of the addition into one of the its input registers, as shown in Fig. 14.2, in order to minimize area occupation;
- multiplexer network to address all data in internal state and from the previous operation to the inputs of the SHA-256 core and adder;
- a Finite State Machine (FSM), which controls the flow of operations, i.e., *instance*, *reseed* and *generate*;

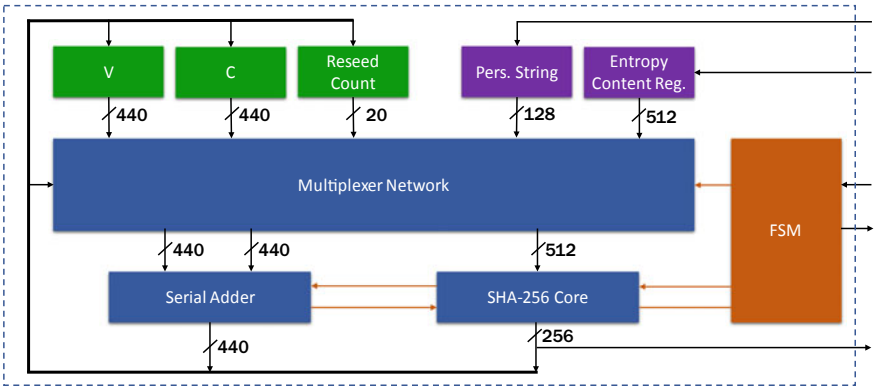

Fig. 14.2 Hash DRBG design architecture developed

- a DRBG self-test module (not present in Fig. 14.2), in order to diagnose possible failures inside the circuitry.

14.4 Results

For the Hash DRBG IP-core characterization, two different technologies have been identified as representative of potential targets for implementations of such hardware accelerator for security applications: Intel Stratix IV FPGA and Silvaco PDK 45 nm Open Cell Library [7] (i.e., ASIC standard-cell technology). In both cases different implementation effort corners were tested, in order to evaluate the trade-off between throughput and area. Concerning the Intel Stratix IV FPGA technology, the synthesis and layout flow performed with high performance constraints gives a maximum operative frequency of 180 MHz, meaning a throughput of 720 Mbps considering the single core instance, for an overall occupation of 5949 ALMs (Adaptive Logic Modules). The implementation on Silvaco ASIC standard-cell is able to reach a throughput even up to 4 Gbps, since the maximum frequency is equal to 1 GHz still for single core version of the IP-core, for a logic complexity of 118.98 kGE corresponding to an area of approximately 0.094 mm^2.

14.5 Conclusions

This paper presented the IP-core design related to a digital Random Number Generator (RNG), one of the most significant part required to implement algorithms for authentication, confidentiality, message integrity and security applications in general.

The proposed architecture is based on one of the Deterministic Random Bit Generators (DRBGs) approved by NIST according to trade-off analysis between throughput, area and security strength. Hash DRBG with SHA-256 as cryptographic core proved to be the most efficient solution in terms of throughput per logic complexity, among the solutions offering maximum security strength (i.e., 256 bits).

The RNG IP-core obtained has been tested by means of NIST Statistical Test Suite, thus stating that the sequences of bits generated cannot be distinguished from a true random sequence of numbers, and therefore validating its use for cryptographic applications. It has been also implemented on FPGA and ASIC standard-cell technologies for characterization. The implementation on Intel Stratix IV FPGA reported a throughput of 720 Mbps at 180 MHz with a maximum occupation of about 6000 ALMs, while the synthesis on Silvaco 45 nm ASIC standard-cell [7] reported a throughput of 4 Gbps at 1 GHz with a maximum logic complexity of about 119 kGE.

References

1. Barker E, Kelsey J (2016) Recommendation for Random Bit Generator (RBG) constructions. Special Publication 800-90C, NIST
2. Barker E, Kelsey J (2015) Recommendation for random number generation using deterministic random bit generators. Special Publication 800-90A, NIST
3. Lo Bello L, Mariani R, Mubeen S, Saponara S (2019) Recent advances and trends in on-board embedded and networked automotive systems. IEEE Trans Ind Inf 15:1038–1051
4. Pelzl J, Paar C (2011) Understanding cryptography. Springer, Berlin
5. Dang QH (2015) Secure hash standard. Technical report, NIST
6. Dichtl M, Golić JD (2007) High speed true random number generation with logic gates only. In: Cryptographic hardware and embedded systems—CHES 2007. Lecture Notes in Computer Science, vol 4727. Springer, Berlin, 45–62
7. Vasyltsov I, Hambardzumyan E, KimBohdan Y-S, Karpinskyy B (2008) Fast digital TRNG based on metastable ring oscillator. In: Cryptographic hardware and embedded systems—CHES 2008. Lecture Notes in Computer Science, vol 5154. Springer, Berlin, 164–180
8. Schmid M (2015) ECDSA—Application and implementation failures
9. Silvaco PDK 45nm Open Cell Library. https://www.silvaco.com/products/nangate/FreePDK45_Open_Cell_Library/index.html