„PSEUDO-RANDOM NUMBERS GENERATORS IMPLEMENTED WITH FPGA TECHNOLOGY"

Cracow Grid Workshop 2017

Marek Wróblewski[1], Marek Sawerwain[1]
[1] Institute of Control & Computation Engineering
University of Zielona Gora
M.Wroblewski@issi.uz.zgora.pl, M.Sawerwain@issi.uz.zgora.pl

Agenda

1. Introduction
2. Pseudo random number generators
3. PRNGs implementations with FPGA
4. Performance details
5. Summary
6. References

Introduction

Generating pseudo-random numbers is a very important task in the context of many computational methods, e.g.

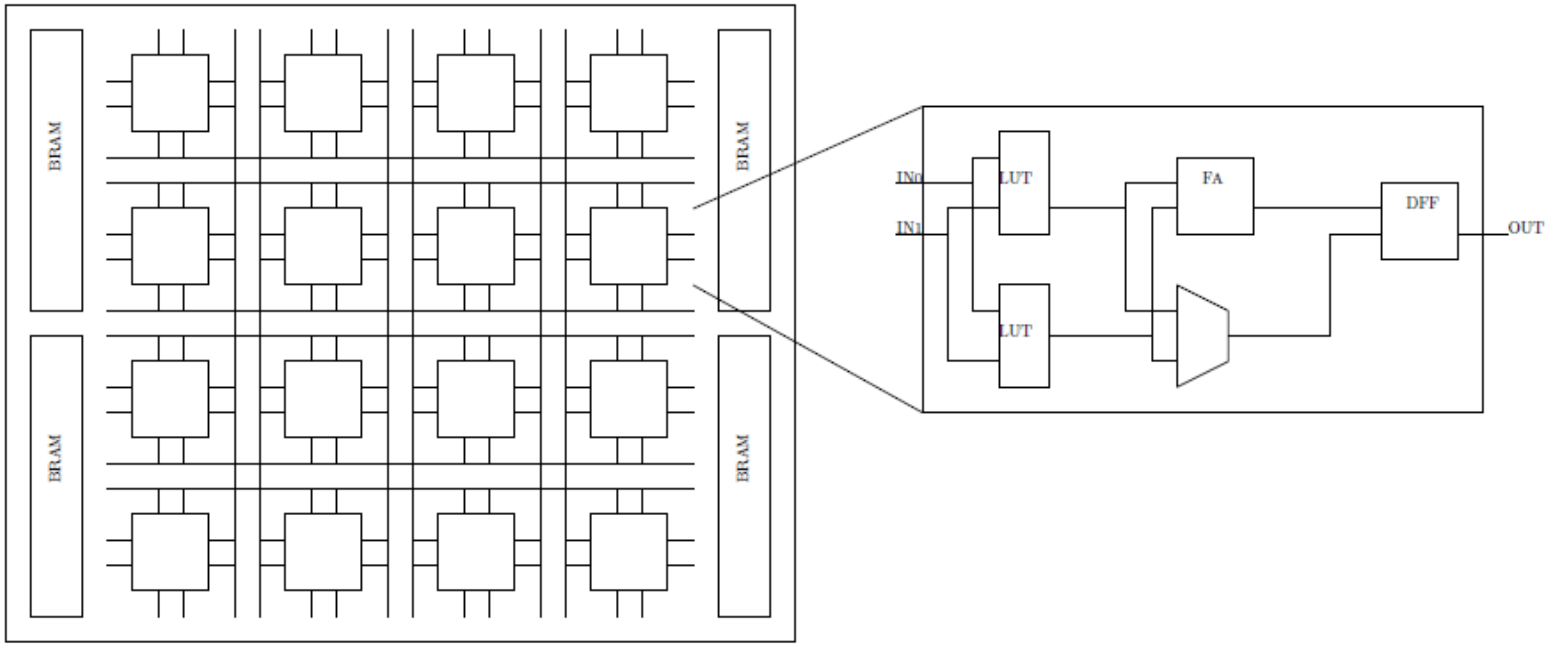
-based on the Monte Carlo method,
-cryptography applications.

Important to know the efficiency and quantity of energy needed to efficiently generate large amounts of value from pseudorandom numbers generators.

Scalable of FPGA technology.

Introduction

The general scheme for building a typical FPGA. The main element is the socalled lookup-table (LUT), which allows you to build any Boolean functions. There are also BRAM - BLOCK RAM memory units that act as fast auxiliary memory, which is capable of working with any clock allowed by a particular FPGA.



Pseudo random number generators

List of periods for individual generators and indication of the most important parameters characterizing a given pseudorandom string generator.

Generator	Parameters	Period
MRG32k3a	—	$3.1 \cdot 10^{57}$
MT19937	—	$4.3 \cdot 10^{6001}$ (or $2^{19937} - 1$)
GM19	k=15, q=28, g=$2^{(19)} - 1$	$2.7 \cdot 10^{11}$
GM31	k=11, q=14, g=$2^{(31)} - 1$	$4.6 \cdot 10^{18}$
GM61	k=24, q=74, g=$2^{(61)} - 1$	$5.3 \cdot 10^{36}$
LFSR113	—	$1.0 \cdot 10^{34}$
XORSHIFT	k=5	2^{160}
MWC256	k=257	2^{8222}
CMWC4096	k=4097	10^{33000}

Pseudo random number generators

Mersenne Twister (MT):

$$x_{k+n} = x_{k+m} \oplus \left(x_k^u :: x_{k+1}^l\right) A \quad .$$

MRG32k3a:

$$X_{n+1} = AX_n (\mathrm{mod}\, m_1),\ Y_{n+1} = BY_n (\mathrm{mod}\, m_2) \quad .$$

LFSR113:

$$X_n = (X_{n-p} + X_{n-p+q}) (\mathrm{mod}\, 2),\ U_n = \sum_{i=1}^{32} X_{ns+i-1} 2^{-i} \quad .$$

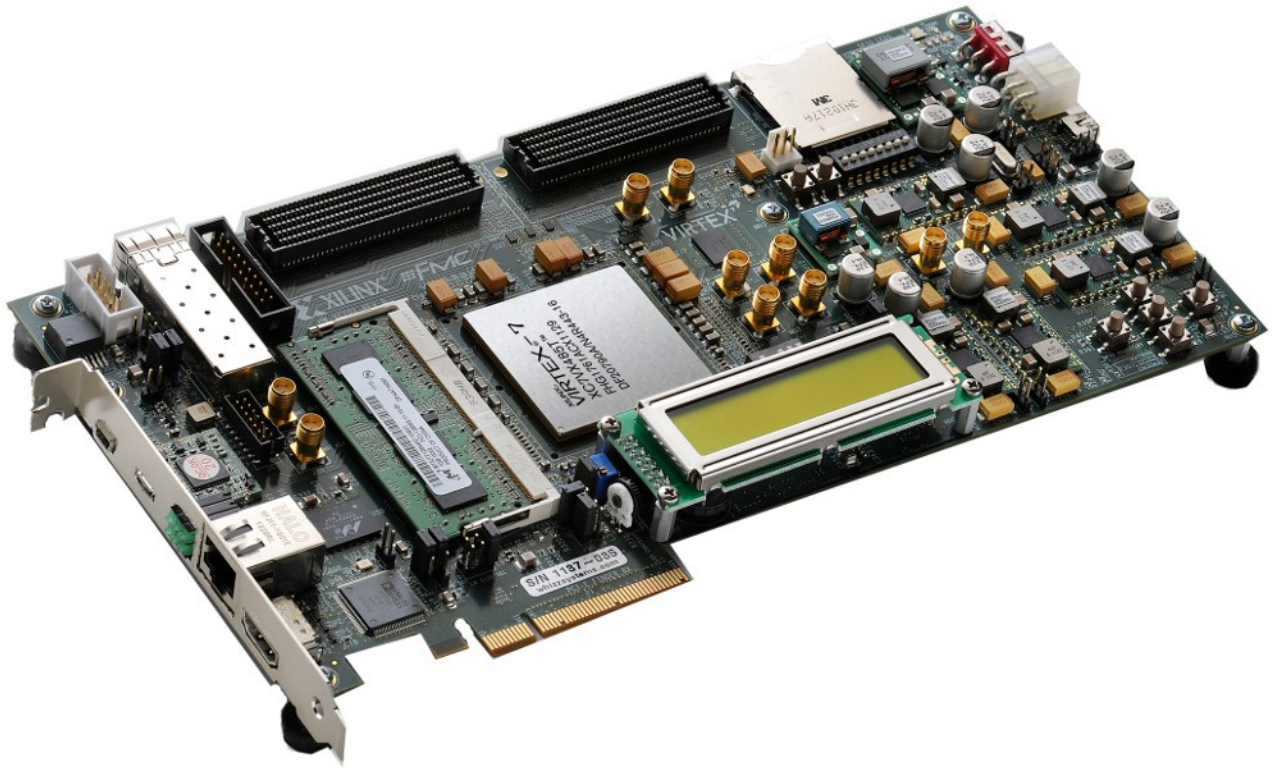
Pseudo random number generators

GM19, GM31 and GM61:

$$\begin{pmatrix} x_i^{(n)} \\ y_i^{(n)} \end{pmatrix} = M \begin{pmatrix} x_i^{(n-1)} \\ y_i^{(n-1)} \end{pmatrix} (\mathrm{mod} g),$$

$$a^{(n)} = \sum_{i=0}^{s-1} \lfloor 2x_i^{(n)}/g \rfloor \cdot 2^i.$$

Xilinx Virtex-7 FPGA VC707



Fot. https://dev.sifive.com

Xilinx Virtex-7 FPGA VC707

Clocking

- Fixed Oscillator with differential 200MHz output used as the "system" clock for the FPGA
- Programmable Oscillator with 156.250 MHz as the default output and frequency targeted for Ethernet applications but oscillator is programmable for many end uses
- Differential SMA clock input
- Differential SMA GTX reference clock input
- Jitter attenuated clock used to support CPRI/OBSAI applications that perform clock recovery from a user-supplied SFP/SFP+ module

Configuration

- Onboard JTAG configuration circuitry to enable configuration over USB
- JTAG header provided for use with Xilinx download cables such as the Platform Cable USB II
- 128MB (1024Mb) Linear BPI Flash for PCIe® Configuration
- 16MB (128Mb) Quad SPI Flash

Expansion Connectors

- FMC1 – HPC (8 XCVR, 160 single ended or 80 differential (34 LA pairs, 24 HA pairs, 22 HB pairs) user-defined pins)
- FMC2 – HPC (8 XCVR, 116 single ended or 58 differential (34 LA pairs, 24 HA pairs) user-defined pins)
- Vadj supports 1.8V
- IIC

Communication & Networking

- Gigabit Ethernet GMII, RGMII and SGMII
- SFP+ transceiver connector
- GTX port (TX, RX) with four SMA connectors
- UART To USB Bridge
- PCI Express x8 gen2 Edge Connector (lay out for Gen3)

Memory

- 1GB DDR3 SODIMM 800MHz / 1600Mbps
- 128MB (1024Mb) Linear BPI Flash for PCIe Configuration
- SD Card Slot
- 8Kb IIC EEPROM

Control & I/O

- 5X Push Buttons
- 8X DIP Switches
- Rotary Encoder Switch (3 I/O)
- AMS FAN Header (2 I/O)

Debug & Analog Input

- 8 GPIO Header, 9 pin removable LCD
- Analog Mixed Signal (AMS) Port

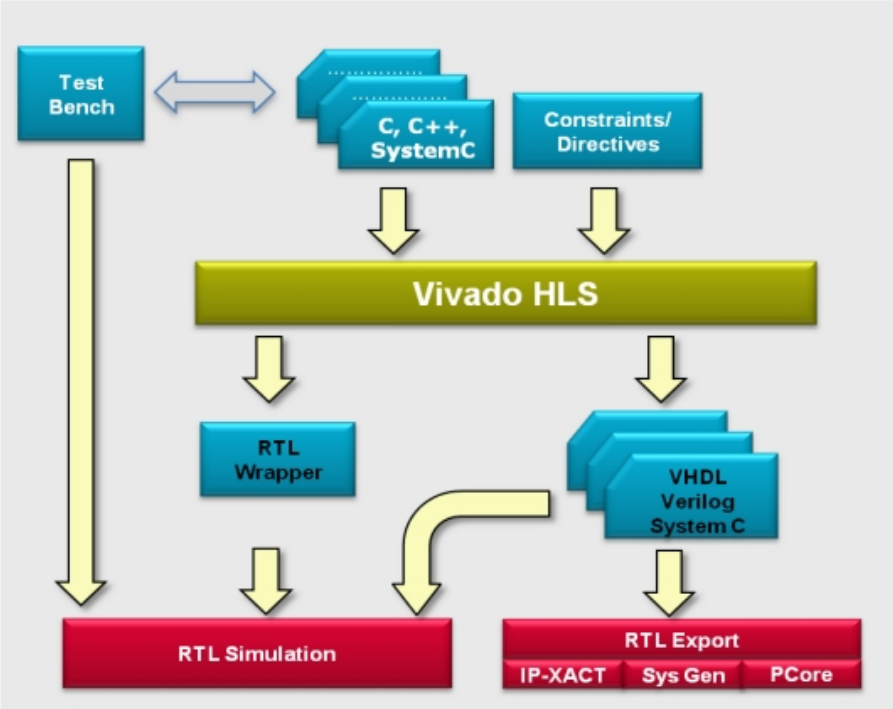
Power

- 12V wall adapter or ATX
- Voltage and Current measurement capability

Display

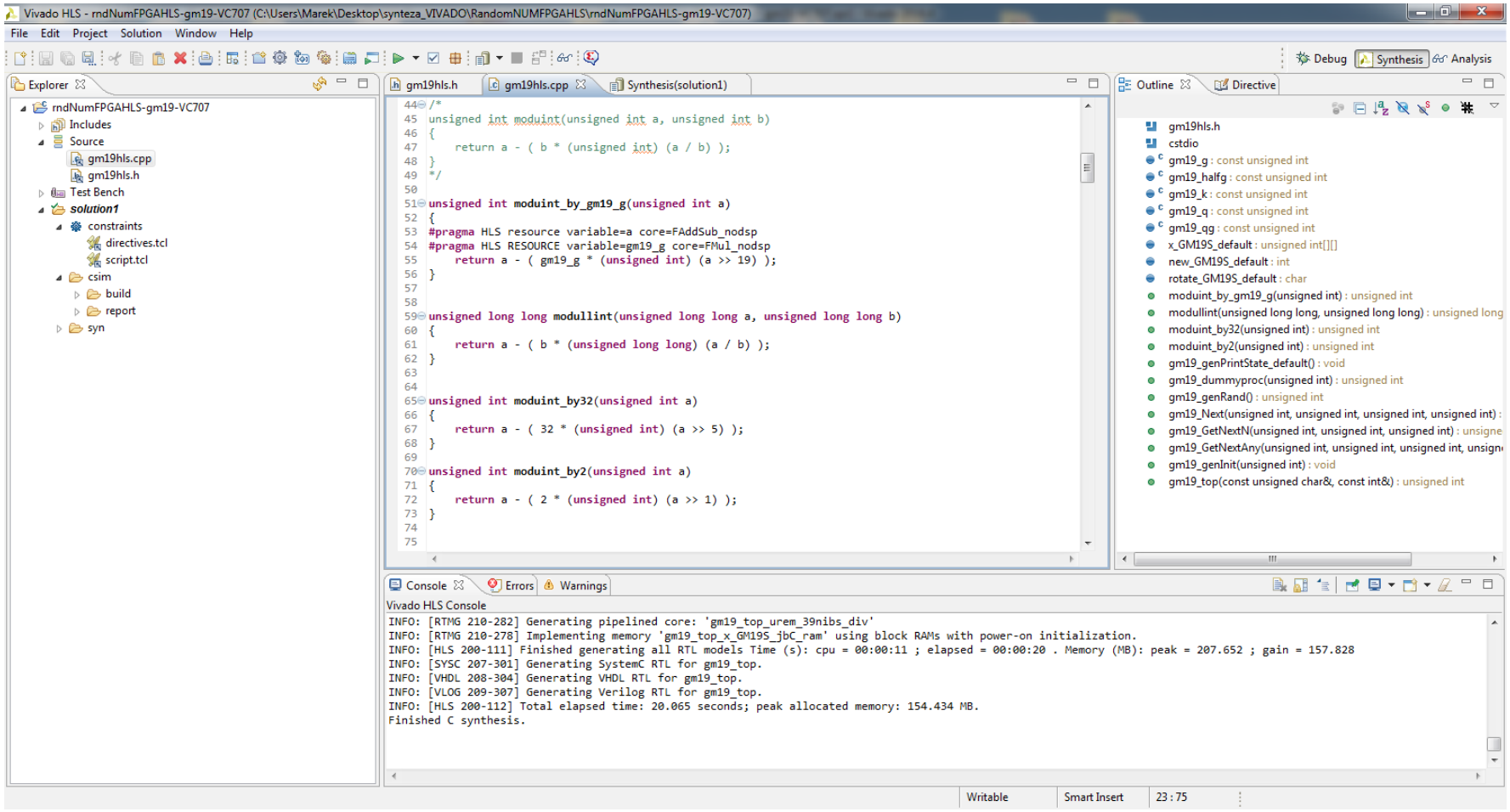
- HDMI Video OUT
- 2 x16 LCD display
- 8X LEDs

Just enter the code in the classic ANSI C language to get a complete conversion and hardware implementation. Automatic implementation of resources in FPGA.

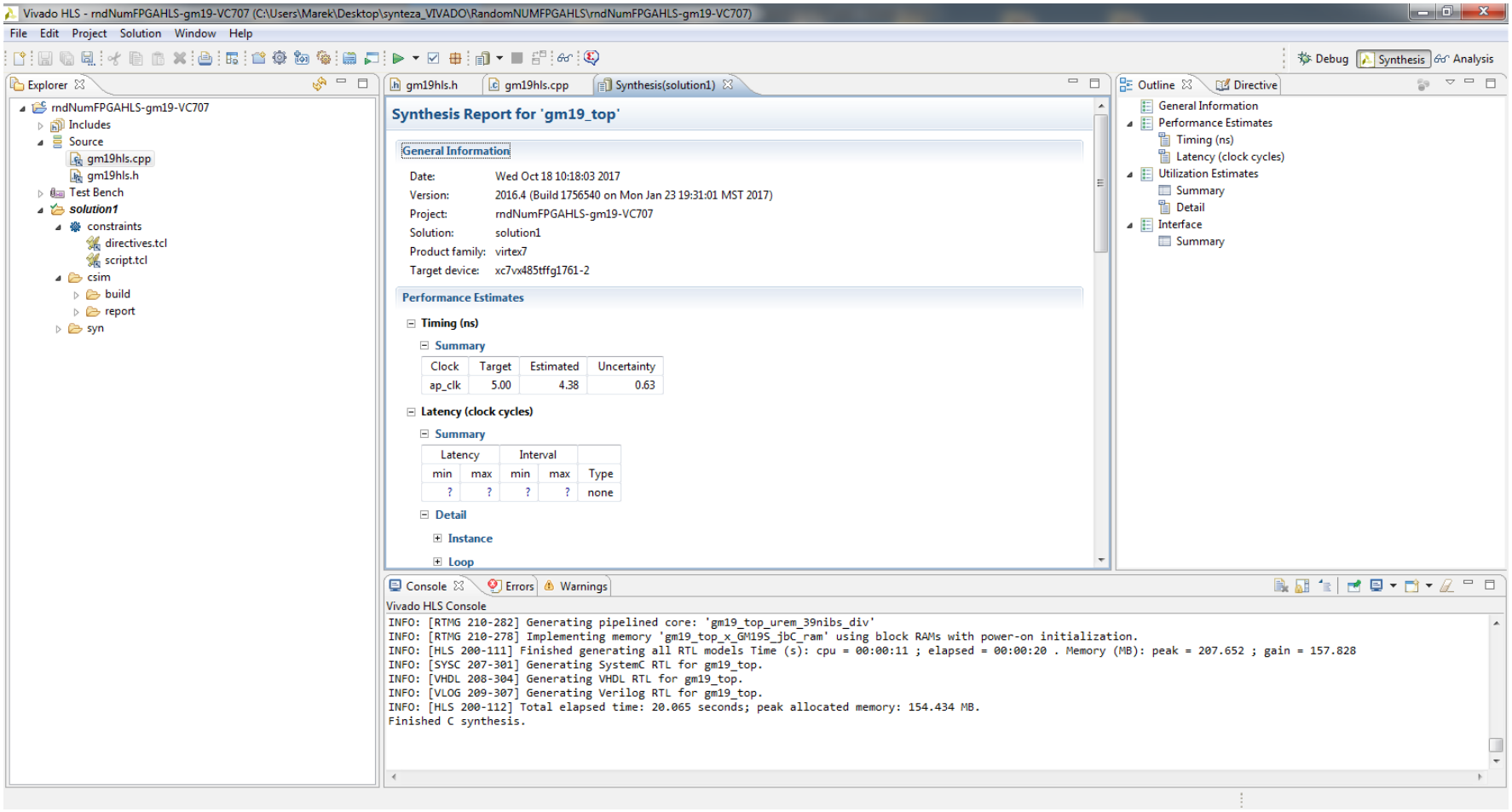


Source: Vivado Design Suite User Guide High-Level Synthesis UG902

PRNGs implementations with FPGA



PRNGs implementations with FPGA



PRNGs implementations with FPGA

In the GM and MT generators for loops use unroll-loop. Optimization technology contribute to increased performance, for example for the GM19 generator, unroll loop expansion can be done manually or use the #pragma HLS UNROLL expression:

```
gm19_genRand_fl1: for(i=0;i<32;i++) {
#pragma HLS UNROLL
   x_GM19S_default[old][i] =
     moduint_by_gm19_g( (gm19_qg+gm19_k*x_GM19S_default[New][i] -
         gm19_q*x_GM19S_default[old][i]) );
}
```

For example, to increase performance in GM generators, modulo division by g is implemented as follows:

```
ullint modullint_by_gm31_g(ullint a) {
    return a - ( gm31\_g * (ullint) (a >> 31) );
}
```

The use of multiplication and bit shift significantly improves performance, "ullint" is a redefinition of an unsigned long long int.

PRNGs implementations with FPGA

The amount of hardware resources (where BRAM means block-ram and FPGA has direct access to it, DSP - computational unit for realisation Arithmetics operations, FF - ip-ops and LUT - Lookup table) of the Virtex 7 VC707 FPGA for the pseudo-random sequence generators used. The parentheses give an approximate percentage as the type of resources used throughout the entire layout used (the total number of individual resources is the last row in the table).

	BRAM18K	DSP48E	FF	LUT
GM19	2 (≈ 0 %)	40 (1 %)	11553 (1 %)	16173 (5 %)
GM31	4 (≈ 0 %)	44 (≈ 1 %)	9422 (≈ 1 %)	10354 (≈ 3 %)
GM61	8 (≈ 0 %)	224 (8 %)	32808 (5 %)	36842 (12 %)
LFSR113	0 (0 %)	0 (0 %)	283 (≈ 0 %)	769 (≈ 0 %)
MGR32K3	0 (0 %)	232 (8 %)	4010 (≈ 0 %)	5070 (1 %)
MT19937	2 (≈ 0 %)	14 (≈ 0 %)	914 (≈ 0 %)	2009 (≈ 0 %)
XORSHIFT	0 (0 %)	4 (≈ 0%)	356 (≈ 0 %)	795 (≈ 0 %)
MWC256	2 (≈ 0 %)	4 (≈ 0%)	411 (≈ 0 %)	4149 (≈ 1 %)
CMWC4096	8 (≈ 0 %)	2 (≈ 0%)	2337 (≈ 0 %)	67663 (≈ 22 %)
Total:	2060	2800	607200	303600

Performance details for GM61

A few remarks:
- 200 MHz clock is used

- The basic version of the code running in the FPGA environment needed 6786 clock cycles.

- The loop for extraction (unroll) resulted in 3329 clock cycles.

- Our own modulo operation resulted in 145 clock cycles.

- Two modifications of the code gave 46 times the acceleration.

- Methods built into the HLS environment are not optimized for performance and hardware resources used by the FPGA.

- One generator needs 0.242W (estimation).

Performance details

Time results for generating 10^8 pseudorandom numbers using the implementation using the Intel i7 4790k 4.0 GHz processor.
In addition to the selected generators, performance results were also presented using SSE intrusions.

Generator	Time in sec.		Numbers per sec.	
	without OPT	opt -O2	without OPT	Z opt -O2
GM19	34,35	8,16	2,910,731	12,242,678
GM19 SSE	1,50	1,37	66,513,021	72,858,343
GM31	42,02	8,18	2,379,779	12,214,872
GM31 SSE	1,96	1,77	50,975,653	56,461,881
GM61	99,18	55,17	1,008,219	1,812,480
GM61 SSE	4,42	4,27	22,601,746	23,417,352
MRG32k3a	2,83	2,10	35,232,516	47,568,946
MRG32k3a SSE	0,49	0,37	200,005,120	265,832,590
MT19937	0,73	0,29	136,274,611	342,655,152
MT19937 SSE	0,27	0,21	366,287,485	474,985,821
LFSR113	0,67	0,31	147,794,142	314,005,657
LFSR113 SSE	0,57	0,56	173,211,521	175,980,059
XORSHIFT	0.30	0.20	333,333,333	500,000,000
MWC256	0.42	0.18	238,095,238	555,555,555
CMWC4096	0.43	0.21	238,095,238	476,190,476

Performance details

The performance of a single generator described by the number of clock (termed clks per num.) necessary to determine the next pseudorandom number and the quantity of random numbers generated per one second (termed PRNs/sec.) assuming that we will use a 200 MHz clock.

Generator	GM19	GM31	GM61
clks per num.	65	160	145
PRNs/sec.	3,076,923	1,250,000	1,379,310
Generator	MRG32k3a	MT19937	LFSR113
clks per num.	99	9242	1
PRNs/sec.	2,020,202	13,497,728	200,000,000
Generator	XORSHIFT	MWC256	CMWC4096
clks per num.	6	8	6
PRNs/sec.	33,333,333	25,000,000	33,333,333

Performance details

The developed RNGFPGLIB library offers access to numerical generators RNGs of the following types: LFSR113, GM19, GM31, GM61, MRG32k3a, MT19937, as well as XORSHIFT, MWC256, CMWC4096.

The library code is based on the code in the original versions presented by the Authors of the individual solutions.

However, to get higher performance or significantly reduce the number of resources modifications are required.

Although it should be noted that the original code can be used as a basis for implementing an algorithm in the FPGA.

However, the performance will be lower.

https://github.com/qMSUZ/PRNGFPGLIB

Summary

1. The performance achieved depends on the type of generator.

2. Ease of implementation of many generators.

3. Ability to scale performance according to the needs of the task.

4. Lower energy consumption for generating a single number.

References

1.Barash, L., Shchur, L.N.: Periodic orbits of the ensemble of Sinai-Arnold cat maps and pseudorandom number generation, Phys. Rev. E 73 (2006) 036701.

2. Barash, L.Yu., Shchur, L.N.: RNGSSELIB: Program library for random number generation, SSE2 realization, Computer Physics Communications, Vol. 182, Issue 7, pp. 15181527 (2011).

3. Barash, L.Yu.: Applying dissipative dynamical systems to pseudorandom number generation: Equidistribution property and statistical independence of bits at distances up to logarithm of mesh size, Europhys. Lett. 95 pp. 10003 (2011).

4. Grassberger, P.: On correlations in good random number generators, Phys. Lett. 181 (1993) 43.

5. L'Ecuyer, P.: Good parameters and implementations for combined multiple recursive random number generators. Operations Research, Vol. 47, Issue 1, pp.159{164 (1999).

6. LEcuyer, P.: Tables of Maximally-Equidistributed Combined LFSR Generators, Math. of Comp., 68(255) (1999) 261269.

7. L'Ecuyer, P.: Random number generation with multiple streams for sequential and parallel computing.In: Proc. of the 2015 Winter Simulation Conf. (WSC), Yilmaz, L., Chan, W.K.V., Moon, I., Roeder, T.M.K., Macal, C., Rossetti, M.D. Eds., Huntington Beach, CA, 2015, pp. 31 { 44 (2015).

Thank you
for your attention