High-Throughput and High-Quality Random Number Generation in FPGAs

Cristian KLEIN

Technical University of Cluj-Napoca

June 24, 2008
1 Motivation

2 Introduction
   - FPGAs
   - Random Numbers

3 Theme of this Thesis

4 TRNG in FPGA
   - Randomness in FPGA
   - RO-based Sampling
   - Fixed-frequency Sampling

5 FloPoCo
   - High-level Goals
   - My Contribution
Motivation

Introduction
- FPGAs
- Random Numbers

Theme of this Thesis

TRNG in FPGA
- Randomness in FPGA
- RO-based Sampling
- Fixed-frequency Sampling

FloPoCo
- High-level Goals
- My Contribution
Motivation

Random numbers are used in:
- cryptography (CryptoRand project)
  - uniformly distributed
  - unpredictable (computationally infeasible to predict)
- simulations (e.g. Monte-Carlo simulations)
  - normally distributed
  - easy to reproduce
2 Introduction

- FPGAs
- Random Numbers

3 Theme of this Thesis

4 TRNG in FPGA

- Randomness in FPGA
- RO-based Sampling
- Fixed-frequency Sampling

5 FloPoCo

- High-level Goals
- My Contribution
FPGAs

- I/O Pad
- Long-distance routing wires
- Switching box
- Local switching box
- Configurable Logic Blocks (CLB)

Diagram showing the layout of a FPGA with various components labeled.
What are random numbers?

```c
int getRandomNumber()
{
    return 4; // chosen by fair dice roll
    // guaranteed to be random.
}
```
For our purpose we will define *random numbers*

- continuous stream of bits
- probability: 1/2 for „0”, 1/2 for „1”
  (independent of previous values)

**Implies**

- uniformly distributed
- unpredictable

**Advantages**

- easy to test statistically (NIST, DieHard, TestU01)
- can be transformed to any distribution
True-Random Number Generators

Based on
- Microscopic entropy: thermal noise, atmospheric noise, atomic decay
- Macroscopic entropy: rolling dices, playing cards etc. (supported by chaos theory)

Pros
- unpredictable

Cons
- sometimes biased (Simple XOR, Neumann)
- low-throughput
“Whoever considers arithmetic for random number generation is in a state of sin.” (John von Neumann)

- FSM with a deterministic evolution
- not really a RNG, but often the output is indistinguishable

**Characteristics**

- much faster than TRNGs
- some are hard to predict (e.g. Blum Blum Shub algorithm)
- the whole output vector depends on an initial seed
Box-Muller Transform

\[ z_0 = \sqrt{-2 \ln u_0} \cdot \sin(2\pi u_1) \]
\[ z_1 = \sqrt{-2 \ln u_0} \cdot \cos(2\pi u_1) \]
1 Motivation

2 Introduction

   • FPGAs
   • Random Numbers

3 Theme of this Thesis

4 TRNG in FPGA

   • Randomness in FPGA
   • RO-based Sampling
   • Fixed-frequency Sampling

5 FloPoCo

   • High-level Goals
   • My Contribution
Theme of this Thesis

1. implement a pure-digital TRNG in FPGA
2. implement an efficient and customisable Box-Muller transform in FPGA
Motivation

Introduction
- FPGAs
- Random Numbers

Theme of this Thesis

TRNG in FPGA
- Randomness in FPGA
- RO-based Sampling
- Fixed-frequency Sampling

FloPoCo
- High-level Goals
- My Contribution
Randomness in FPGA

![Diagram of random number generation in FPGAs]

Cristian KLEIN (UTCN)
RO-based Sampling

Months of efforts, no quality random numbers
Fixed-frequency Sampling (1)

- Requires a *resilience* function to obtain a pure-random stream
Fixed-frequency Sampling (2)
High-Speed TRNG

Ring Oscillator

Ring Oscillator

Ring Oscillator

Ring Oscillator

clk

D Q

Sampling FF

Resilience Function

RandomBit

RandomBit

D Q

Sampling FF

Resilience Function
Results

<table>
<thead>
<tr>
<th>$d$</th>
<th>$r$</th>
<th>$n$</th>
<th>$l$</th>
<th>throughput (Kbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>20</td>
<td>3</td>
<td>12500</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>10</td>
<td>3</td>
<td>6250</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>10</td>
<td>3</td>
<td>3125</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>195</td>
</tr>
</tbody>
</table>

Where:

$2^d$ clock divisor

$2^r$ number of bits of the resilience function

$n$ number of RO

$l$ length of RO
Motivation

Introduction
- FPGAs
- Random Numbers

Theme of this Thesis

TRNG in FPGA
- Randomness in FPGA
- RO-based Sampling
- Fixed-frequency Sampling

FloPoCo
- High-level Goals
- My Contribution
High-level Goals

Easy to create arithmetic operators for FPGA
- with tunable input / output precision
- automatic pipelining
- fine-tuned for several FPGA targets

Easy to combine and reuse in more complex designs
- e.g. reuse shifter and int multiplier in FP multiplier
- automatic pipeline adjustment
- automatic precision augmentation

Automatic test-case generation
- operator depending testing
- also allow random / exhaustive testing
- validated using software arithmetic
My Contribution to FloPoCo

- integrated HOTBM (useful for generating some fixed-point functions)
- integrated FPLog
- FloFP (class simulation FloPoCo operands in software)
- automatic test bench generation

**Box-Muller transform:**

\[
\begin{align*}
    z_0 &= \sqrt{-2 \ln u_0} \cdot \sin(2\pi u_1) \\
    z_1 &= \sqrt{-2 \ln u_0} \cdot \cos(2\pi u_1)
\end{align*}
\]
Conclusions

TRNG on FPGAs
- I have studied two TRNG designs:
  - one does not generate quality random numbers
  - the other works perfectly
- I have also created a high-speed version TRNG for FPGA
- Part of this work has been accepted for being published at ICCP 2008.

FPGA arithmetics
- I have contributed to the FloPoCo project
- paved the way to implement arbitrary-precision Box-Muller transform on FPGAs
Questions?

Thank you for your attention.
Cristian KLEIN.