A Highly-Portable True Random Number Generator based on Coherent Sampling

2019 International Conference on Field-Programmable Logic and Applications

Adriaan Peetermans, Vladimir Rožić and Ingrid Verbauwhede

September 10, 2019

KU LEUVEN

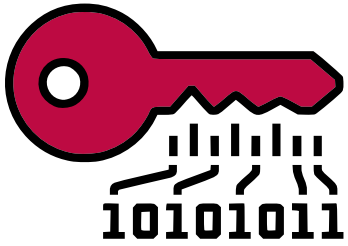
fwo

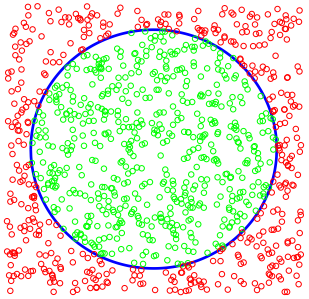
COSIC

Random numbers

How are they used?

- Cryptography:
  - ★ Symmetric key
  - ★ Public key
  - ★ Challenge-response protocols
  - ★ Padding value
  - ★ Masking
- Statistical simulations
  - ★ Monte Carlo
  - ★ Optimisation
  - ★ Initialisation
- Gambling/games
  - ★ Card shuffling
  - ★ Dice throw
  - ★ Roulette

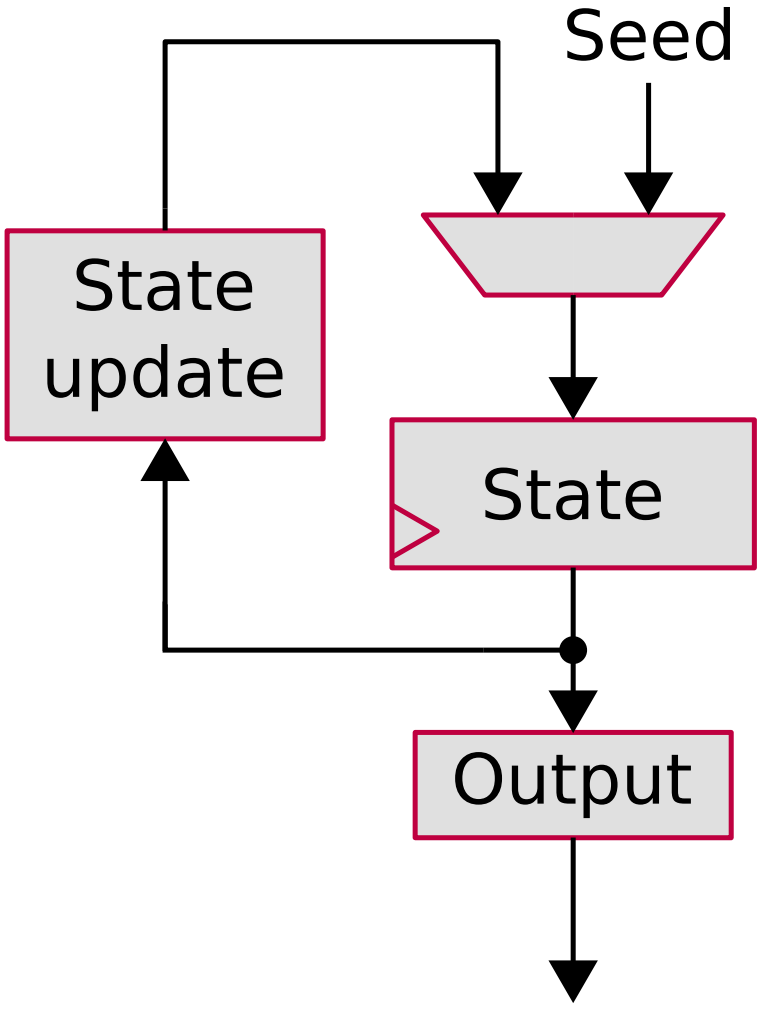




Random numbers

How are they generated?

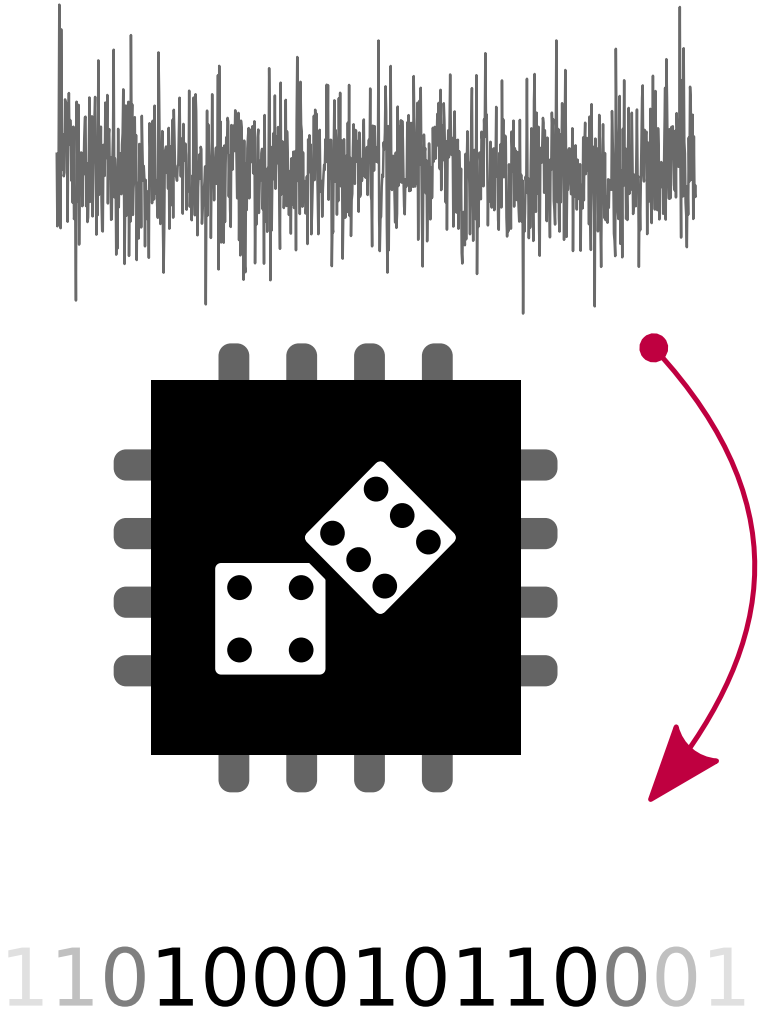
- Pseudo Random Number Generator (PRNG)
  - ★ Deterministic finite state machine expanding the initial seed value
- True Random Number Generator (TRNG)



Random numbers

How are they generated?

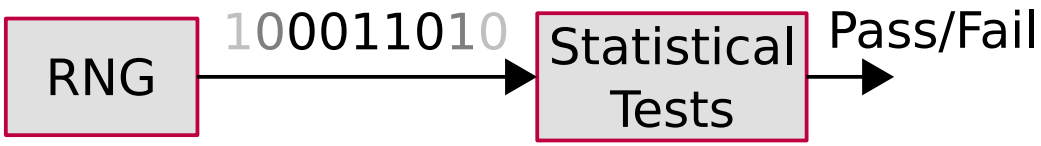
- Pseudo Random Number Generator (PRNG)
- True Random Number Generator (TRNG)
  - ★ Convert electrical noise to digital bitstream
  - ★ Must be accompanied by a stochastic model



Stochastic model

How to make sure the process is truly random?

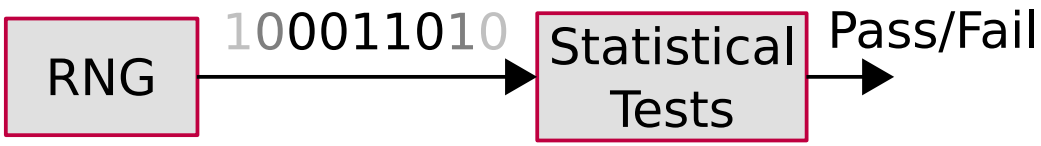
- Old approach:



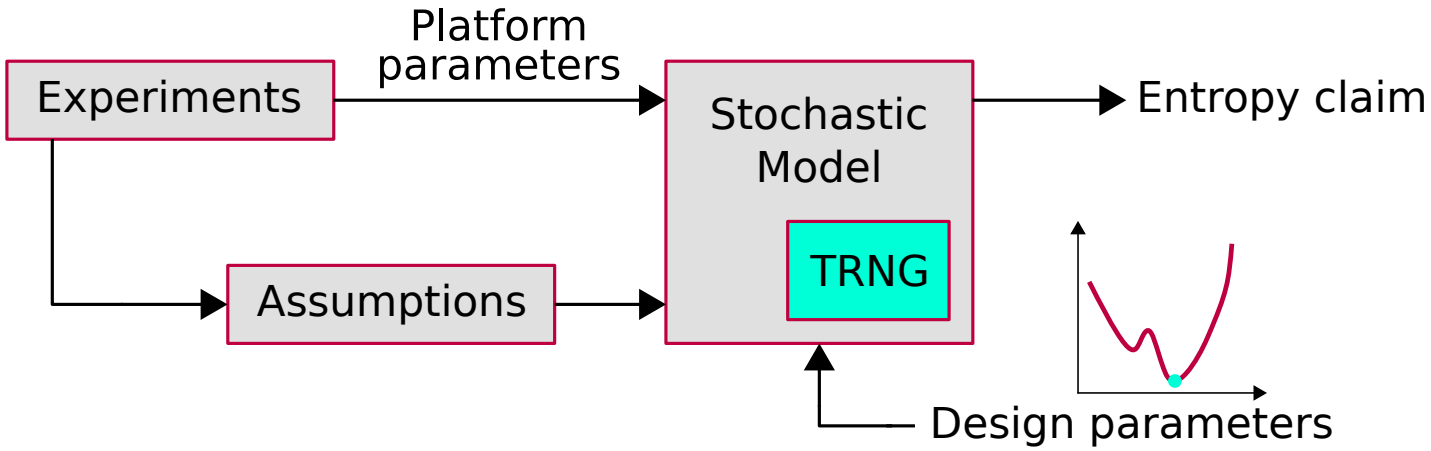
Stochastic model

How to make sure the process is truly random?

- Old approach:



- New approach:



TRNGs for FPGA

TRNGs for FPGA with associated stochastic model:[1]

TRNG type	Area (LUT/Reg)	Power cons. [mW]	Bit rate [Mbit/s]	Feasib. & Repeat.
ERO	46/19	2.16	0.0042	5
COSO	18/3	1.22	0.54	1
MURO	521/131	54.72	2.57	4
PLL	34/14	10.6	0.44	3
TERO	39/12	3.312	0.625	1
STR	346/256	65.9	154	2

[1] O. Petura, et al. "A survey of AIS-20/31 compliant TRNG cores suitable for FPGA devices," in FPL 2016.

TRNGs for FPGA

TRNGs for FPGA with associated stochastic model:[1]

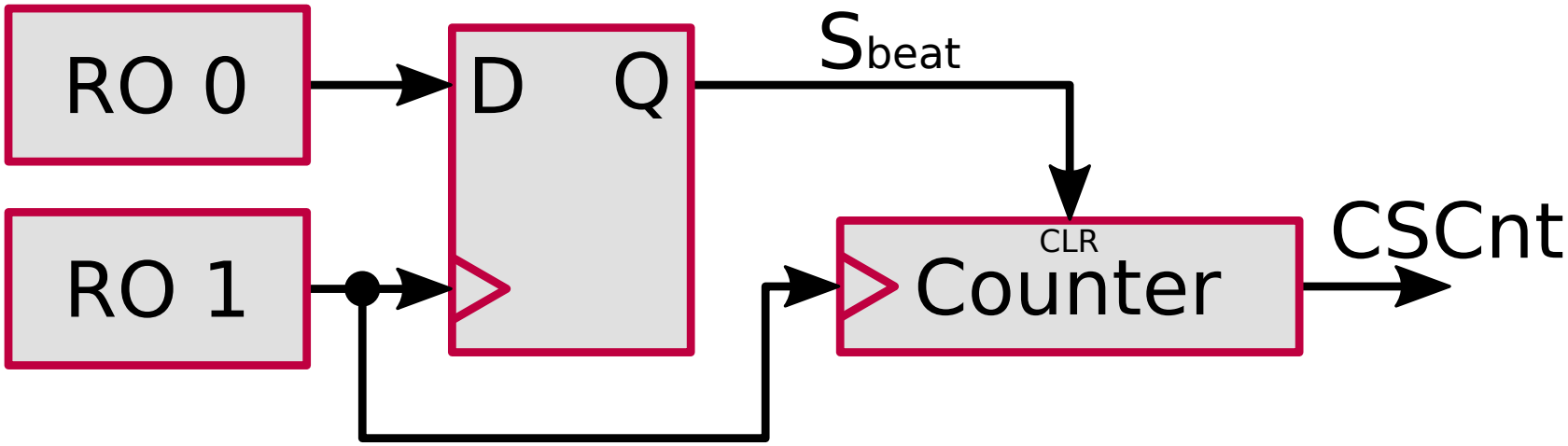
TRNG type	Area (LUT/Reg)	Power cons. [mW]	Bit rate [Mbit/s]	Feasib. & Repeat.
ERO	46/19	2.16	0.0042	5
COSO	18/3	1.22	0.54	1
MURO	521/131	54.72	2.57	4
PLL	34/14	10.6	0.44	3
TERO	39/12	3.312	0.625	1
STR	346/256	65.9	154	2

[1] O. Petura, et al. "A survey of AIS-20/31 compliant TRNG cores suitable for FPGA devices," in FPL 2016.

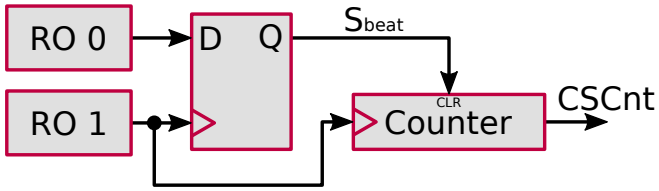
COherent Sampling ring Oscillator (COSO) based TRNG

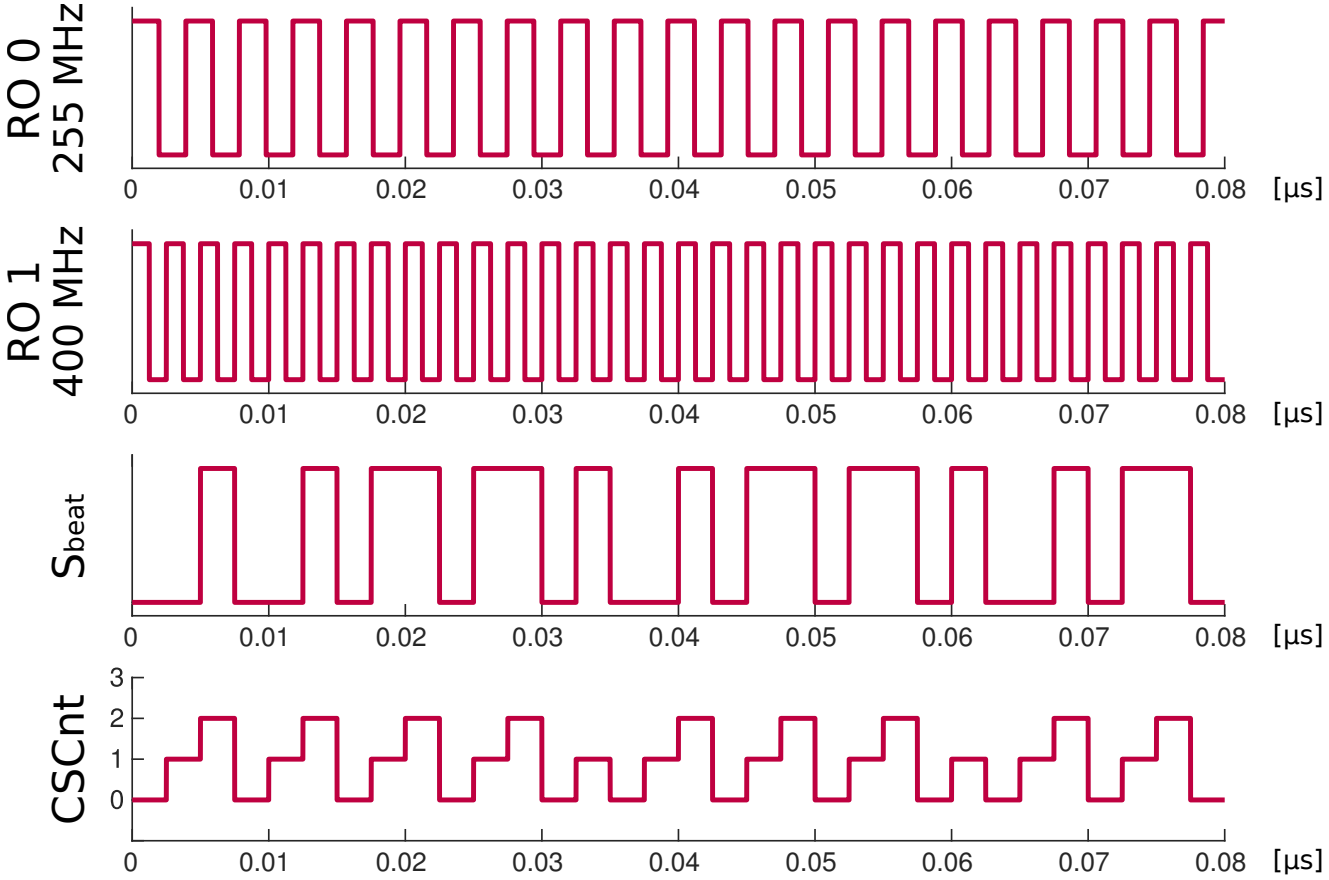
What makes this TRNG hard to implement?

- General architecture:
  - ★ RO 1 samples RO 0
  - ★ Sampling generates low frequency beat signal (S_{beat})
  - ★ Count period length of S_{beat} and reset every negative edge of S_{beat}

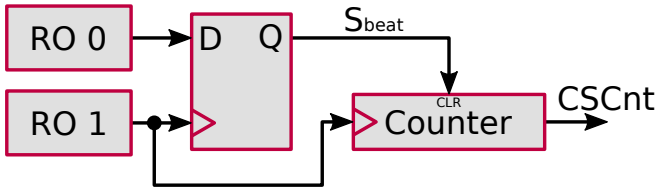


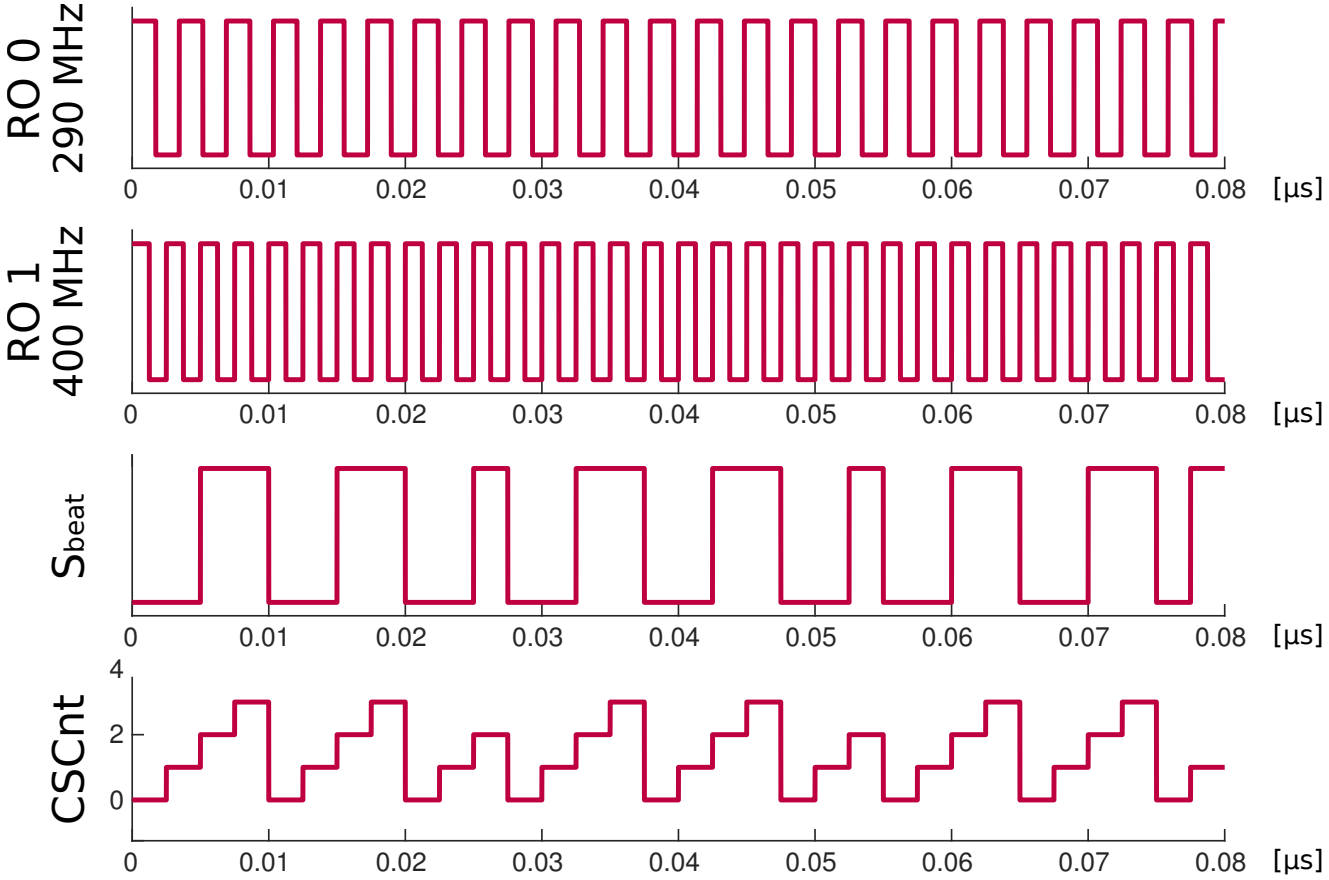
COherent Sampling ring Oscillator (COSO) based TRNG



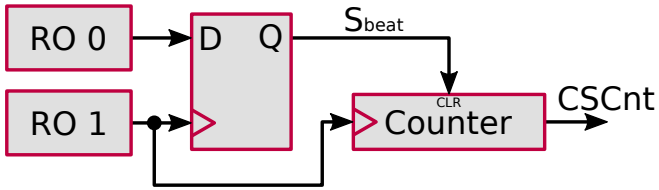


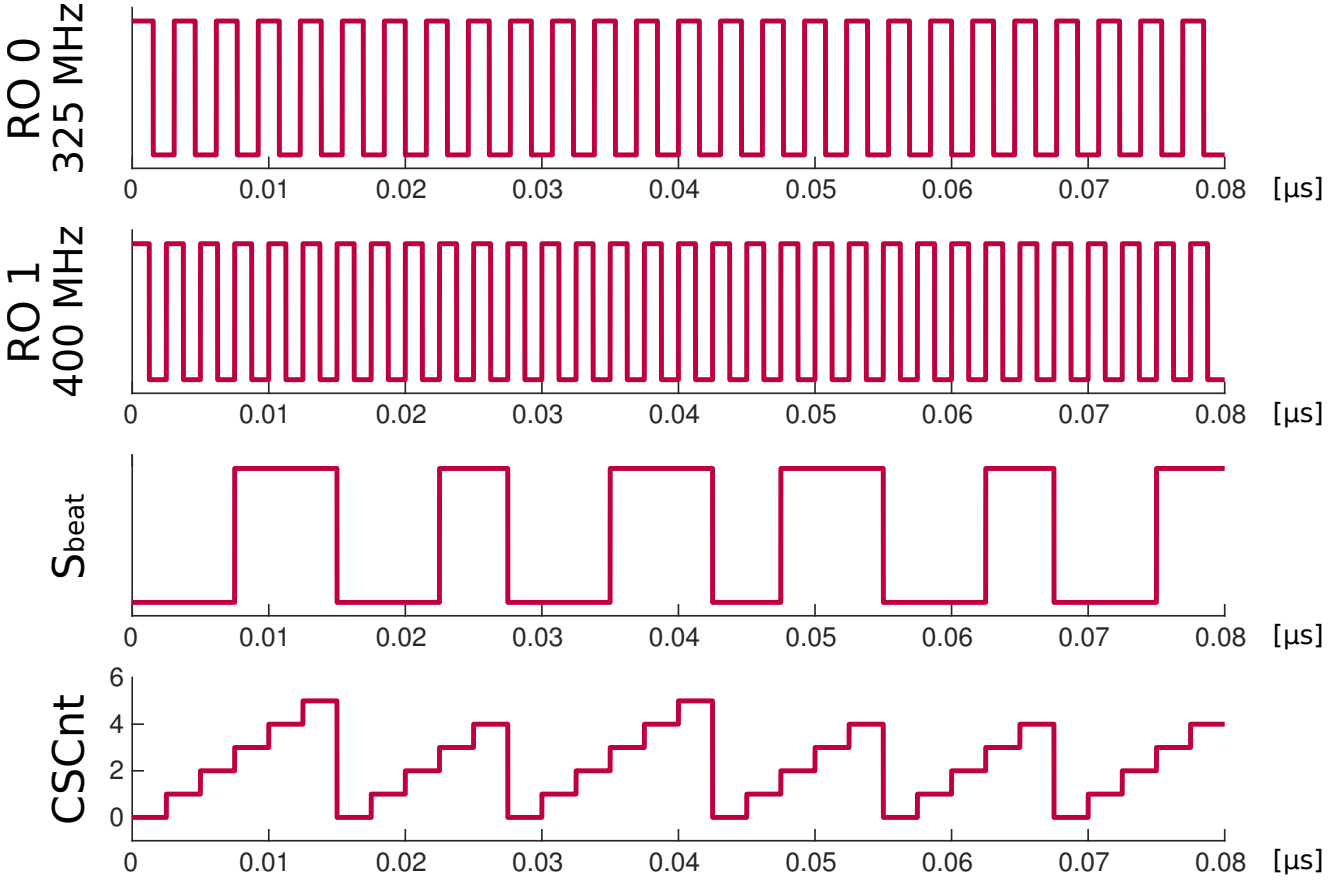
COherent Sampling ring Oscillator (COSO) based TRNG



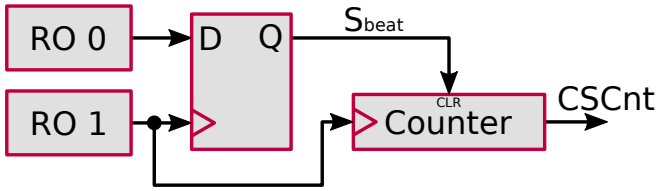


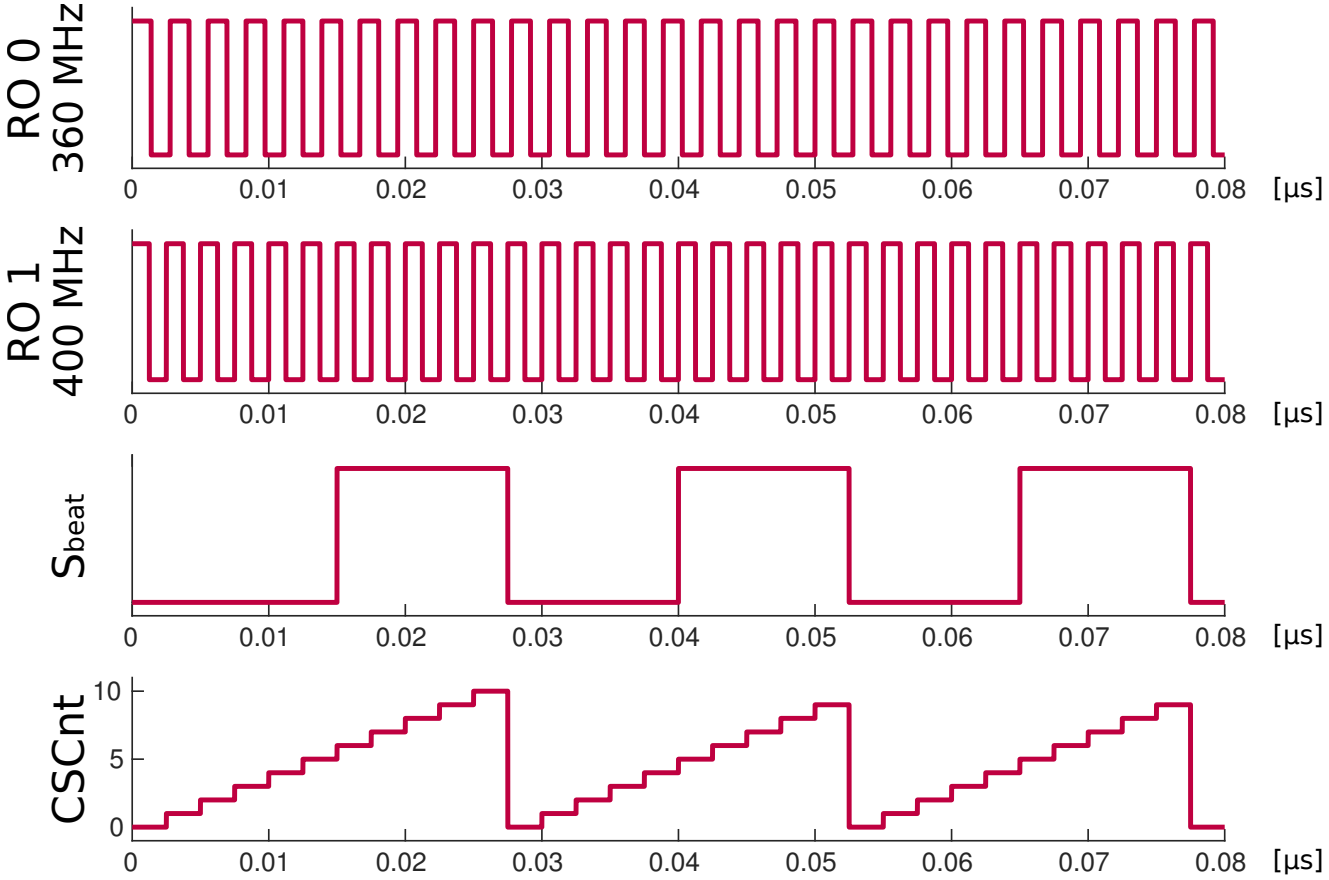
COherent Sampling ring Oscillator (COSO) based TRNG



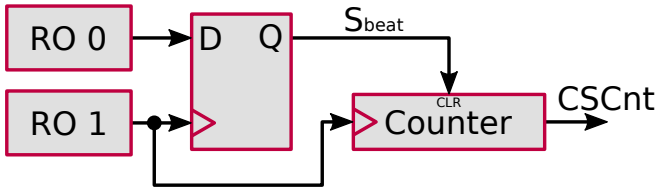


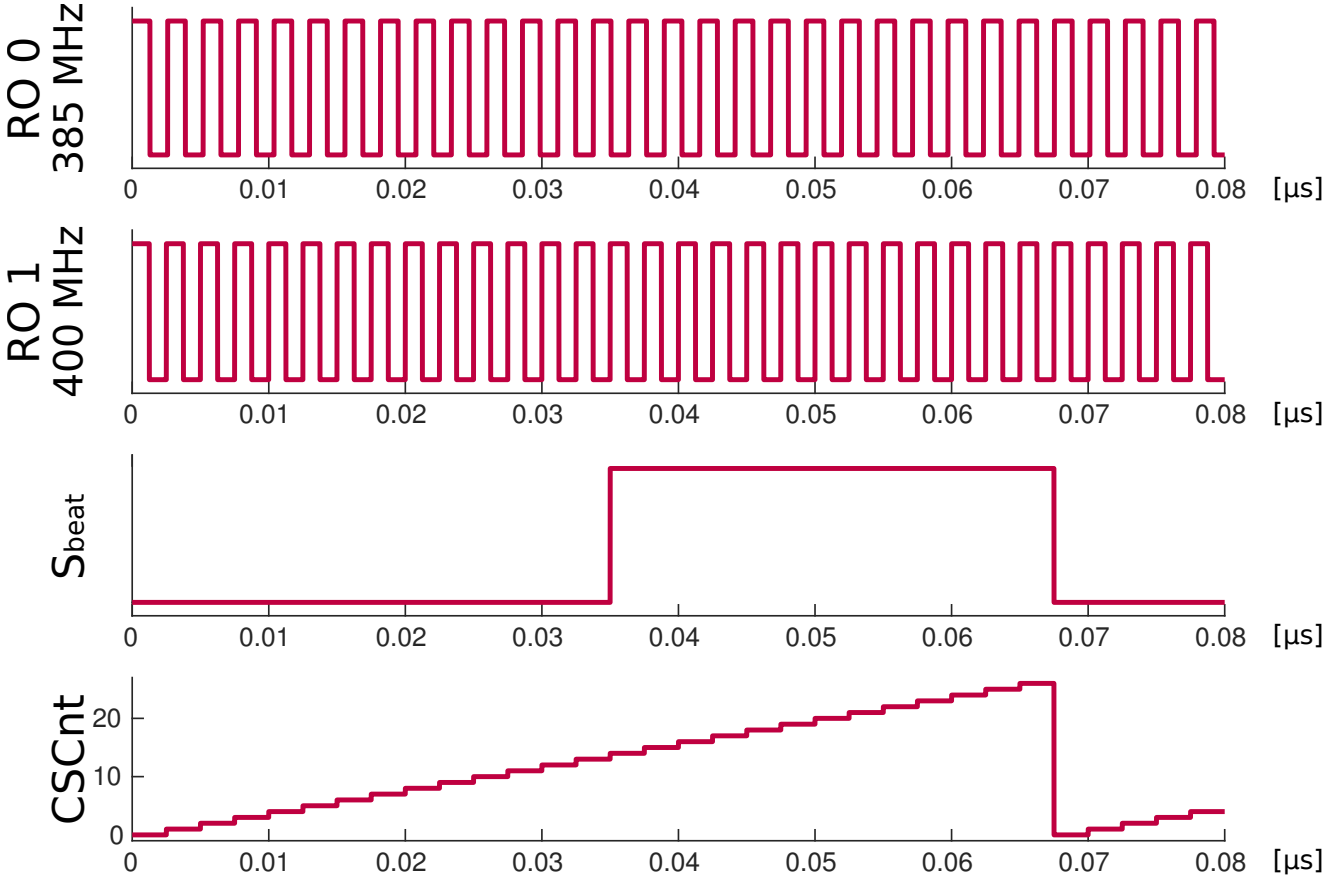
COherent Sampling ring Oscillator (COSO) based TRNG



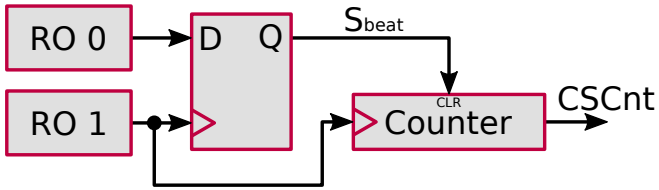


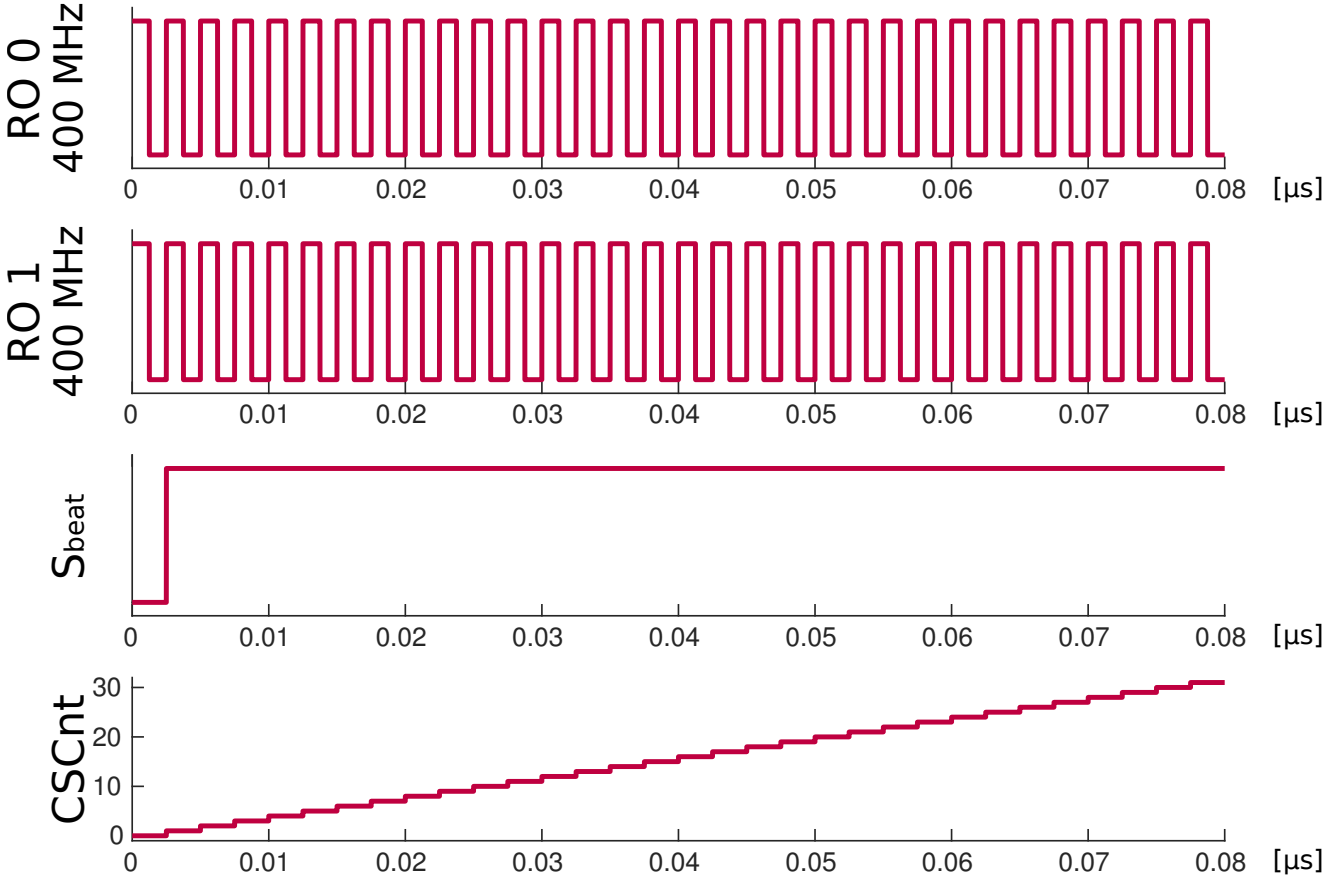
COherent Sampling ring Oscillator (COSO) based TRNG





COherent Sampling ring Oscillator (COSO) based TRNG





COSO stochastic model

Due to jitter in both ROs, *CSCnt* is a discrete random variable:

$$\mathrm{E}[CSCnt] = \frac{\mathrm{E}[T_{RO0}]}{\mathrm{E}[\Delta]}$$

$$\mathrm{Var}[CSCnt] = \mathrm{E}[CSCnt]\frac{\mathrm{Var}[\Delta]}{\mathrm{E}[\Delta]^2}$$

Δ is equal to the period difference of the two ROs:

$$\mathrm{E}[\Delta] = |\mathrm{E}[T_{RO1}] - \mathrm{E}[T_{RO0}]|$$

$$\mathrm{Var}[\Delta] = \mathrm{Var}[T_{RO0}] + \mathrm{Var}[T_{RO1}]$$

Use the LSB of the generated *CSCnt* value as a random bit

COSO stochastic model

Due to jitter in both ROs, *CSCnt* is a discrete random variable:

$$\mathrm{E}[CSCnt] = \frac{\mathrm{E}[T_{RO0}]}{\mathrm{E}[\Delta]}$$

$$\mathrm{Var}[CSCnt] = \mathrm{E}[CSCnt]\frac{\mathrm{Var}[\Delta]}{\mathrm{E}[\Delta]^2}$$

Δ is equal to the period difference of the two ROs:

$$\mathrm{E}[\Delta] = |\mathrm{E}[T_{RO1}] - \mathrm{E}[T_{RO0}]|$$

$$\mathrm{Var}[\Delta] = \mathrm{Var}[T_{RO0}] + \mathrm{Var}[T_{RO1}]$$

Use the LSB of the generated *CSCnt* value as a random bit

COSO stochastic model

Due to jitter in both ROs, *CSCnt* is a discrete random variable:

$$\mathrm{E}[CSCnt] = \frac{\mathrm{E}[T_{RO0}]}{\mathrm{E}[\Delta]}$$

$$\mathrm{Var}[CSCnt] = \mathrm{E}[CSCnt]\frac{\mathrm{Var}[\Delta]}{\mathrm{E}[\Delta]^2}$$

Δ is equal to the period difference of the two ROs:

$$\mathrm{E}[\Delta] = |\mathrm{E}[T_{RO1}] - \mathrm{E}[T_{RO0}]|$$

$$\mathrm{Var}[\Delta] = \mathrm{Var}[T_{RO0}] + \mathrm{Var}[T_{RO1}]$$

Use the LSB of the generated *CSCnt* value as a random bit

COSO stochastic model

Due to jitter in both ROs, *CSCnt* is a discrete random variable:

$$\mathrm{E}[CSCnt] = \frac{\mathrm{E}[T_{RO0}]}{\mathrm{E}[\Delta]}$$

$$\mathrm{Var}[CSCnt] = \mathrm{E}[CSCnt]\frac{\mathrm{Var}[\Delta]}{\mathrm{E}[\Delta]^2}$$

Δ is equal to the period difference of the two ROs:

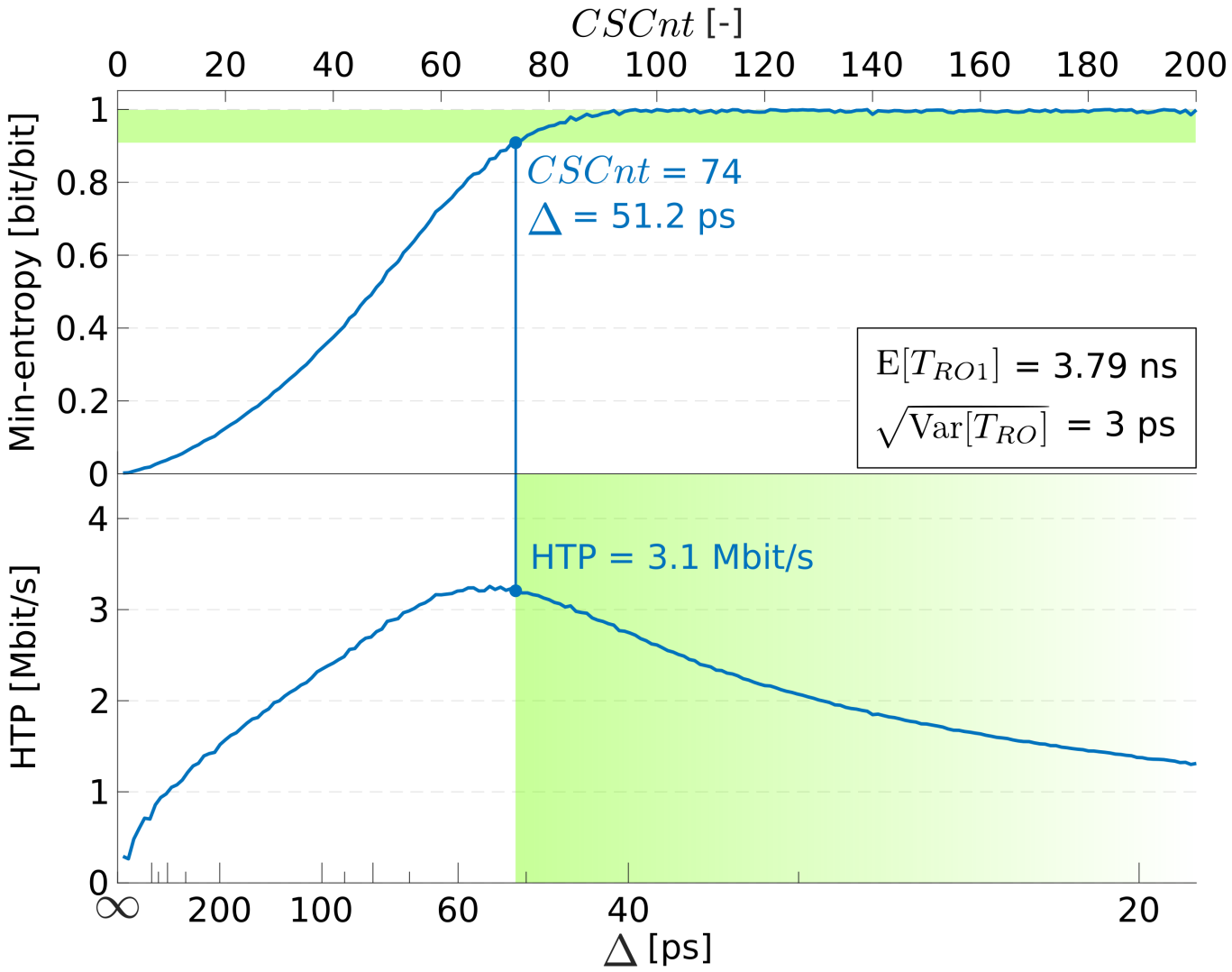
$$\mathrm{E}[\Delta] = |\mathrm{E}[T_{RO1}] - \mathrm{E}[T_{RO0}]|$$

$$\mathrm{Var}[\Delta] = \mathrm{Var}[T_{RO0}] + \mathrm{Var}[T_{RO1}]$$

Use the LSB of the generated *CSCnt* value as a random bit

COSO stochastic model

Entropy versus throughput trade-off:



RO matching in FPGA

How to achieve RO matching in FPGA?

- Search for locations that gives the required matching
  - ★ Slow and labour intensive process
  - ★ Has to be repeated for every device separately, even for the same FPGA family/vendor

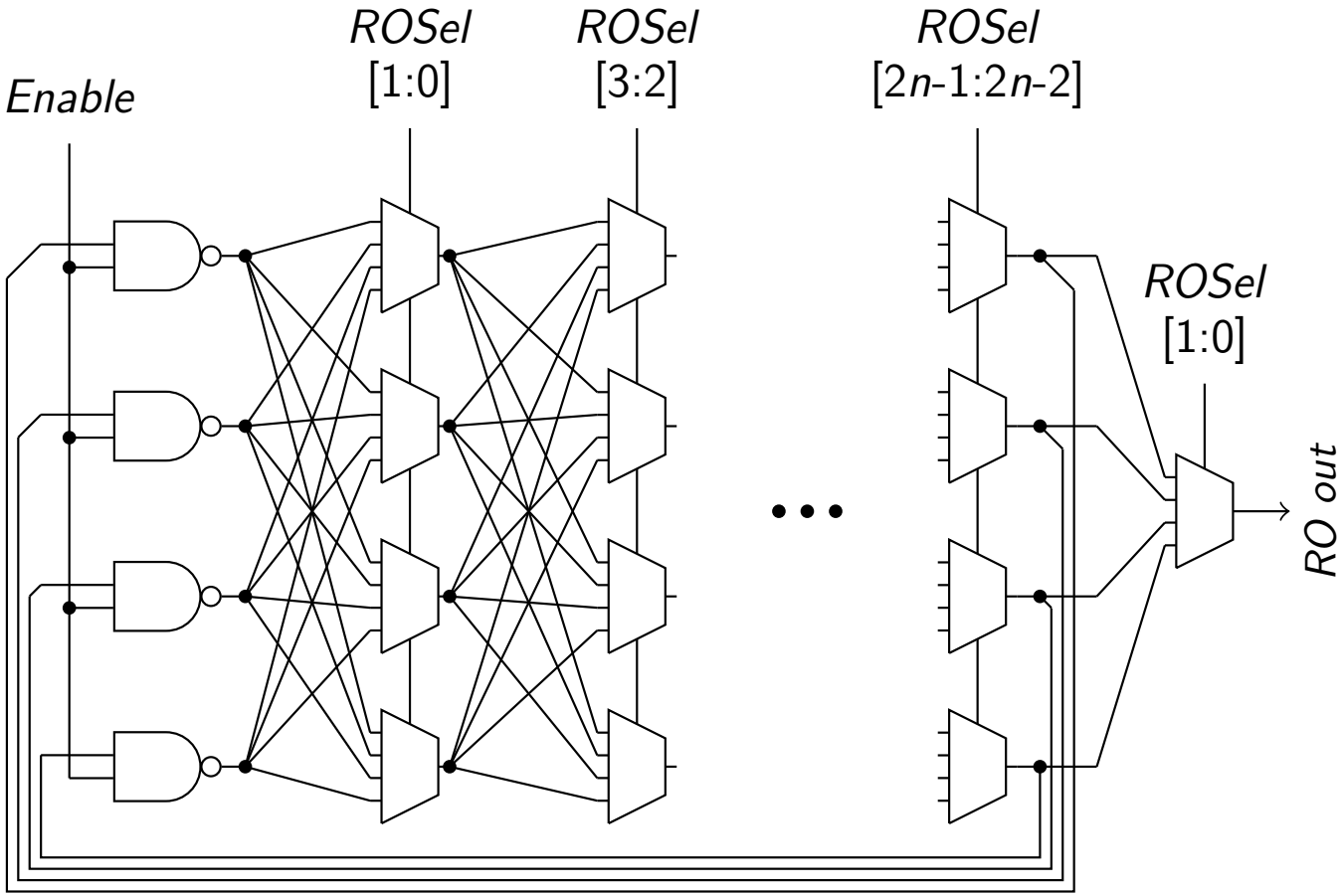
Architecture	FPGA family	Area [DFFs/LUTs]	Throughput [Mbit/s]	Statistical test	Design effort
Original COSO [15]	*Spartan 6*	3/18	0.54	AIS-31 T8	MP
	Cyclone V	3/13	1.44	AIS-31 T8	MP
	SmartFusion2	3/23	0.328	AIS-31 T8	MP
DC-TRNG [18]	*Spartan 6*	128 slices	1.1	AIS-31 T6-T8	MP
	Cyclone V	273 ALMs	1.116	AIS-31 T6-T8	MP & MR
PLL-TRNG [18]	*Spartan 6*	190 slices	1.0416	AIS-31 T6-T8	PLL required
	Cyclone V	273 ALMs	1.04	AIS-31 T6-T8	PLL required
ES-TRNG [11]	*Spartan 6*	5/10	1.15	AIS-31 T0-T5	MP
	Cyclone V	6/10	1.067	AIS-31 T0-T5	MP
TERO [15]	*Spartan 6*	12/39	0.625	AIS-31 T8	MP & MR
	Cyclone V	12/46	1	AIS-31 T8	MP & MR
	SmartFusion2	12/46	1	AIS-31 T8	MP & MR
STR [15]	*Spartan 6*	256/346	154	AIS-31 T8	MP & MR
	Cyclone V	256/352	245	AIS-31 T8	MP & MR
	SmartFusion2	256/350	188	AIS-31 T8	MP & MR

RO matching in FPGA

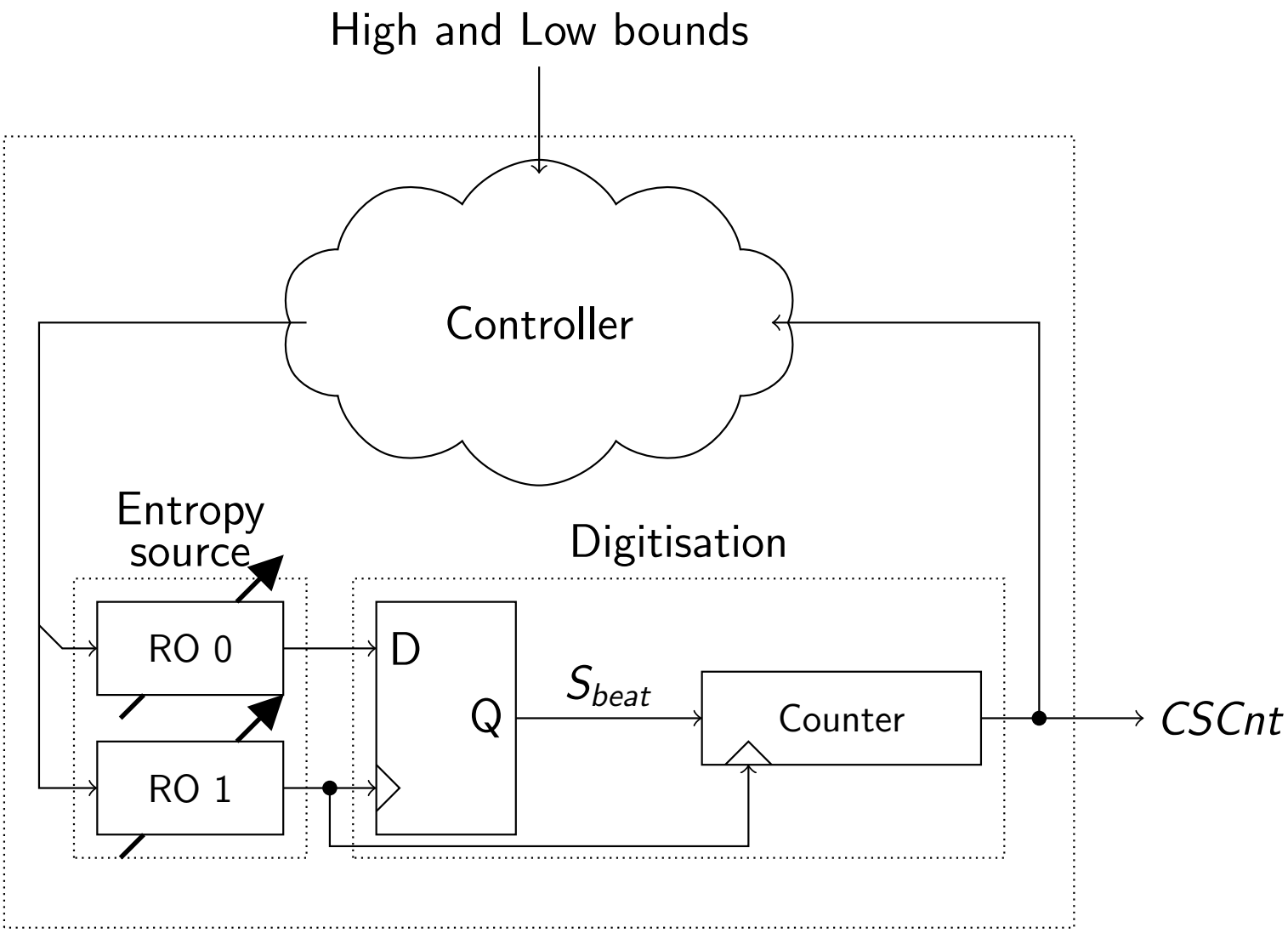
How to achieve RO matching in FPGA?

- Create a reconfigurable RO that can match itself using a feedback mechanism
  - ★ No manual intervention needed
  - ★ Same bitstream can be used for all devices
  - ★ Porting process greatly simplified
  - ★ Control circuit can actively monitor TRNG health and change configuration when needed

Configurable RO



Controller feedback



Controller feedback

Input: $CSCnt$, req

Output: $ROSel$, $matched$

Global constant: L, H

```
goodSamples ← 0, sampleCnt ← 0
ROSel ← 0, matched ← 0
while true do
    if req then
        if L ≤ CSCnt < H then
            goodSamples ← goodSamples + 1
            matched ← 1
        if sampleCnt == 2^7 − 1 then
            if goodSamples == 0 then
                ROSel ← ROSel + 1
                matched ← 0
            goodSamples ← 0
        sampleCnt ← sampleCnt + 1
```

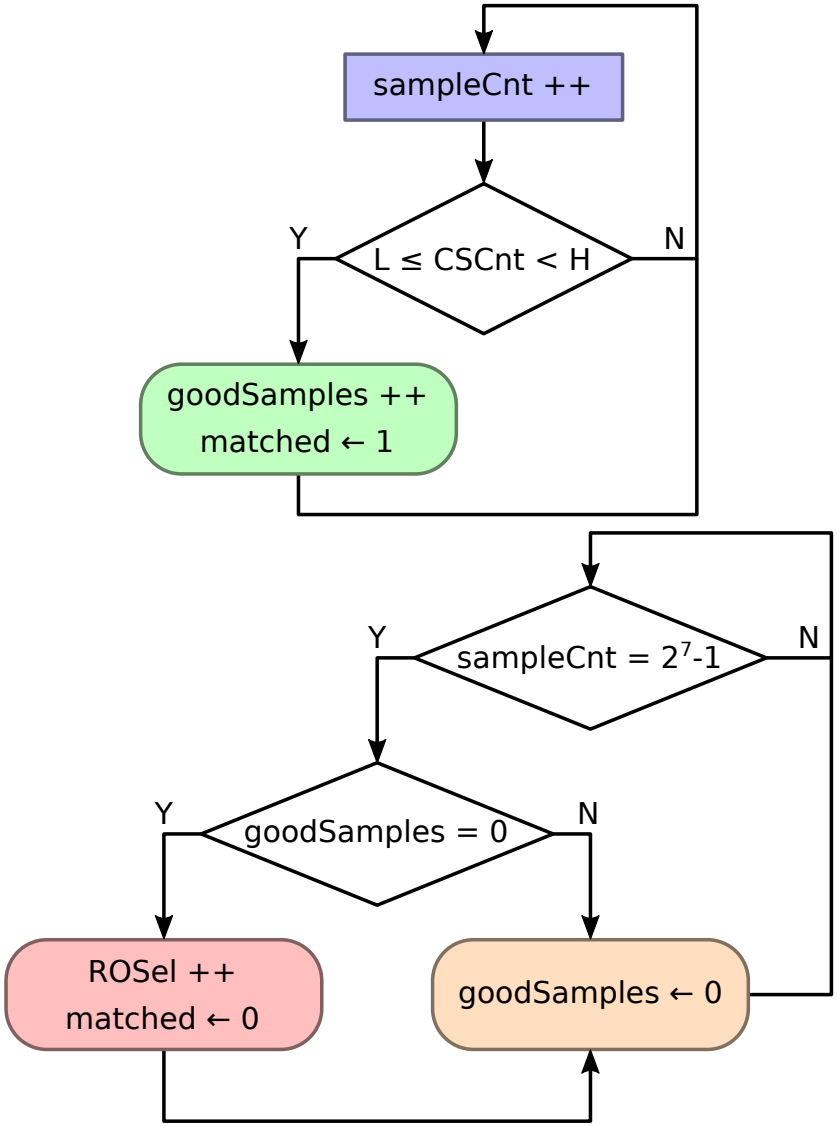
Controller feedback

Input: *CSCnt*, *req*

Output: *ROSel*, *matched*

Global constant: *L*, *H*

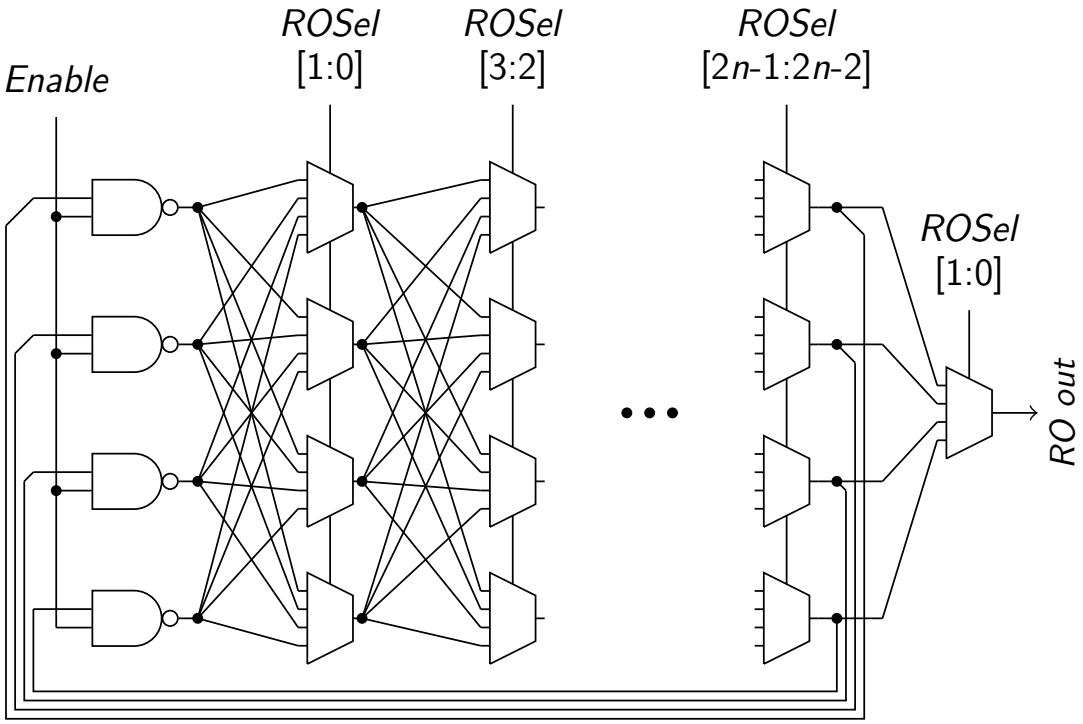
```
goodSamples ← 0, sampleCnt ← 0
ROSel ← 0, matched ← 0
while true do
    if req then
        if L ≤ CSCnt < H then
            goodSamples ← goodSamples + 1
            matched ← 1
        if sampleCnt == 2^7 − 1 then
            if goodSamples == 0 then
                ROSel ← ROSel + 1
                matched ← 0
            goodSamples ← 0
        sampleCnt ← sampleCnt + 1
```



Experimental validation

Experiments should answer the following questions:

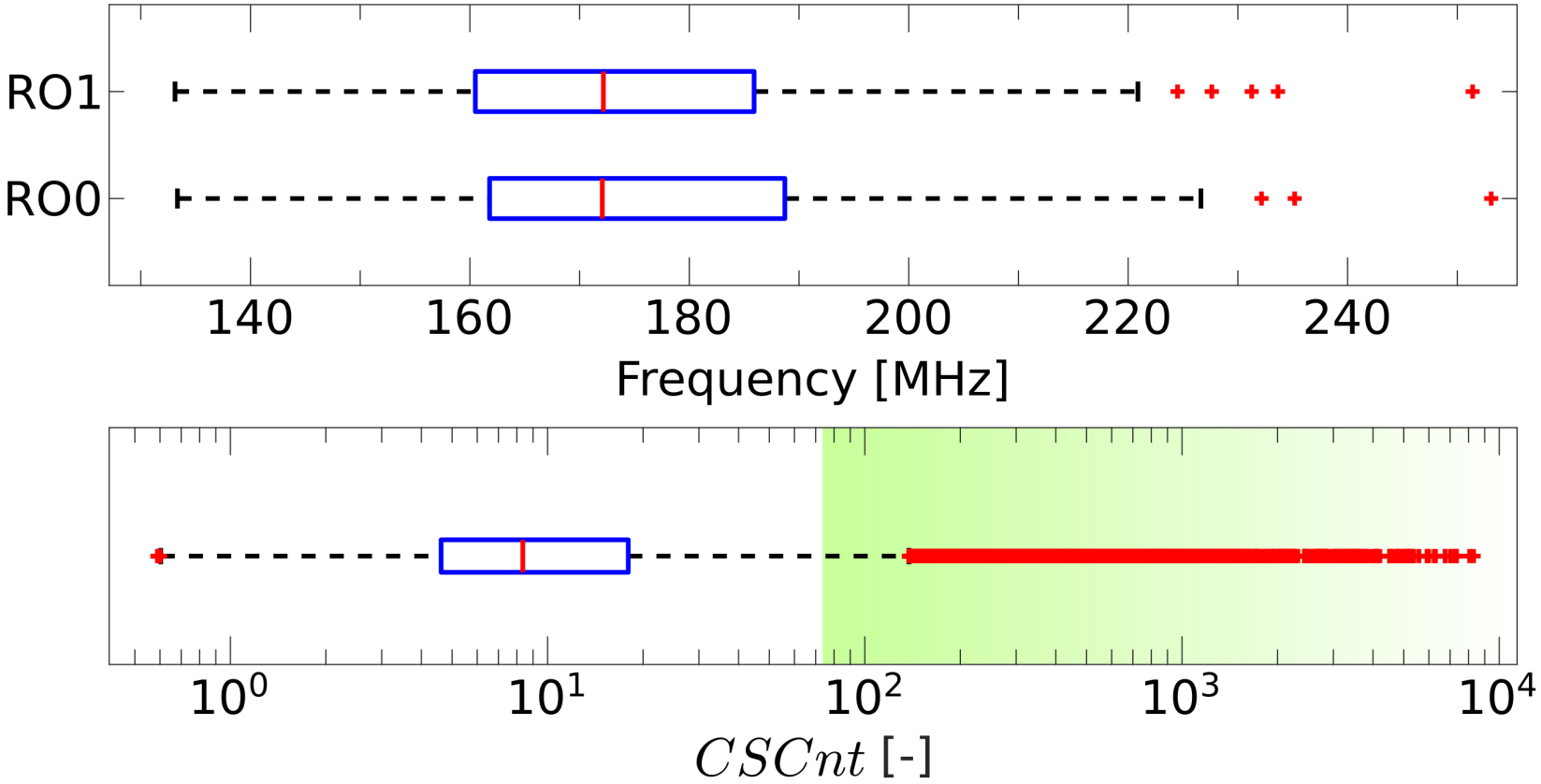
- Can the configurable RO produce a wide range of frequencies?
- Is searching for an optimal placement still necessary?
- Are placement constraints still necessary?
- Can this configurable RO architecture also work on other FPGAs?
- How many stages are necessary?



Feasibility of the architecture

Can the configurable RO produce a wide range of frequencies?

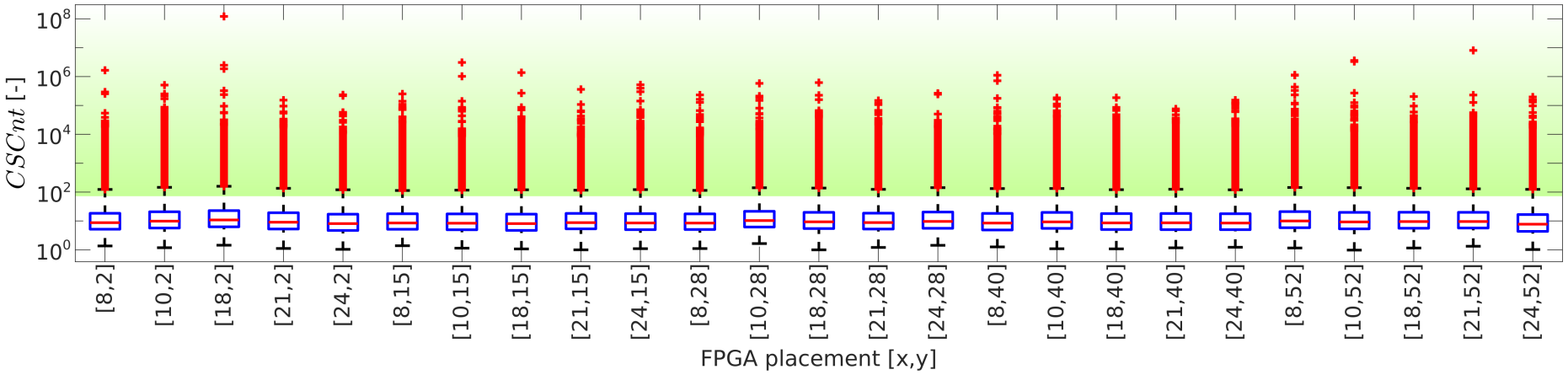
- Spartan 6 results:



Global placement

Is searching for an optimal placement still necessary?

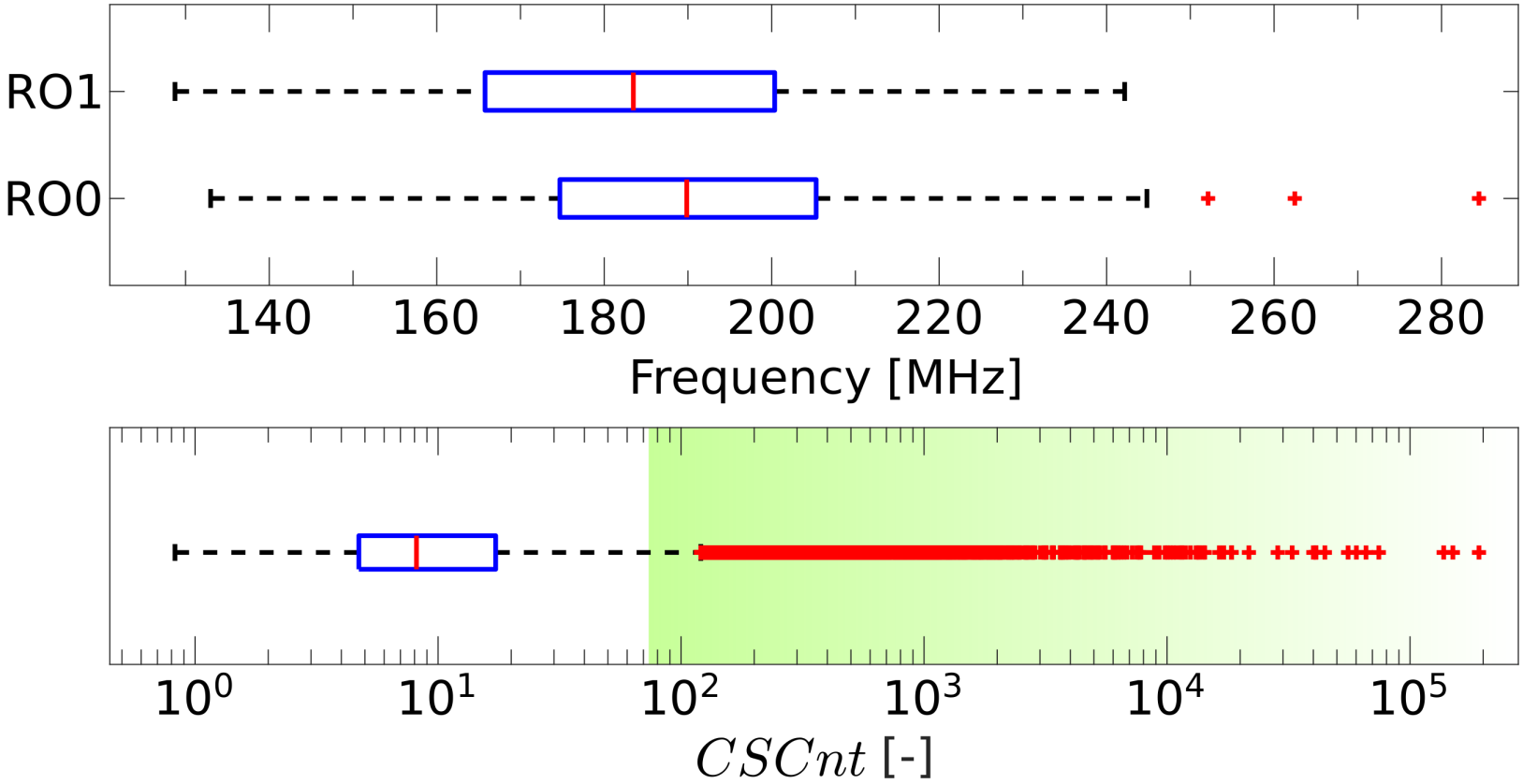
- Spartan 6 results:



Local placement

Are placement constraints still necessary?

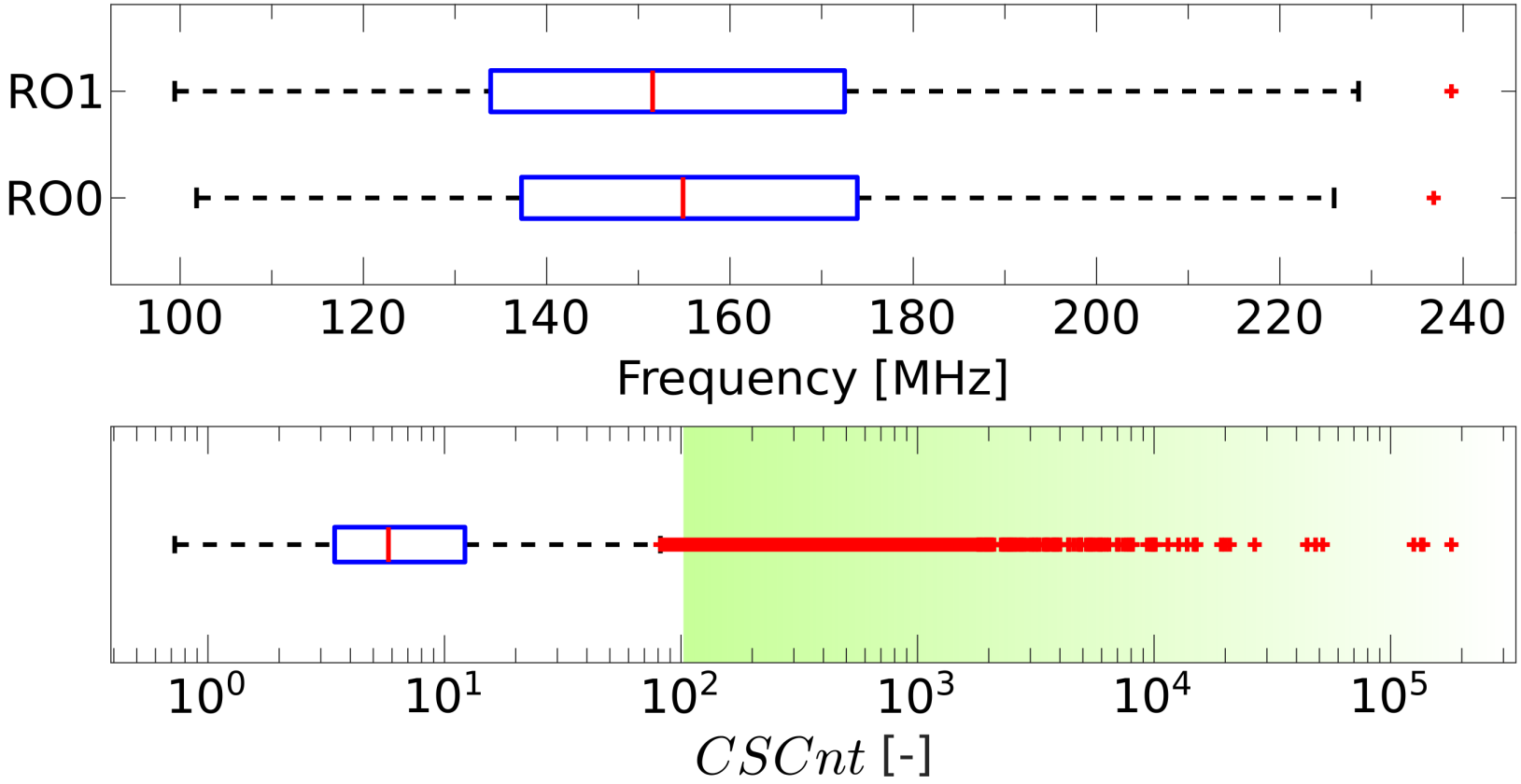
- Spartan 6 results:



Portability

Can this configurable RO architecture also work on other FPGAs?

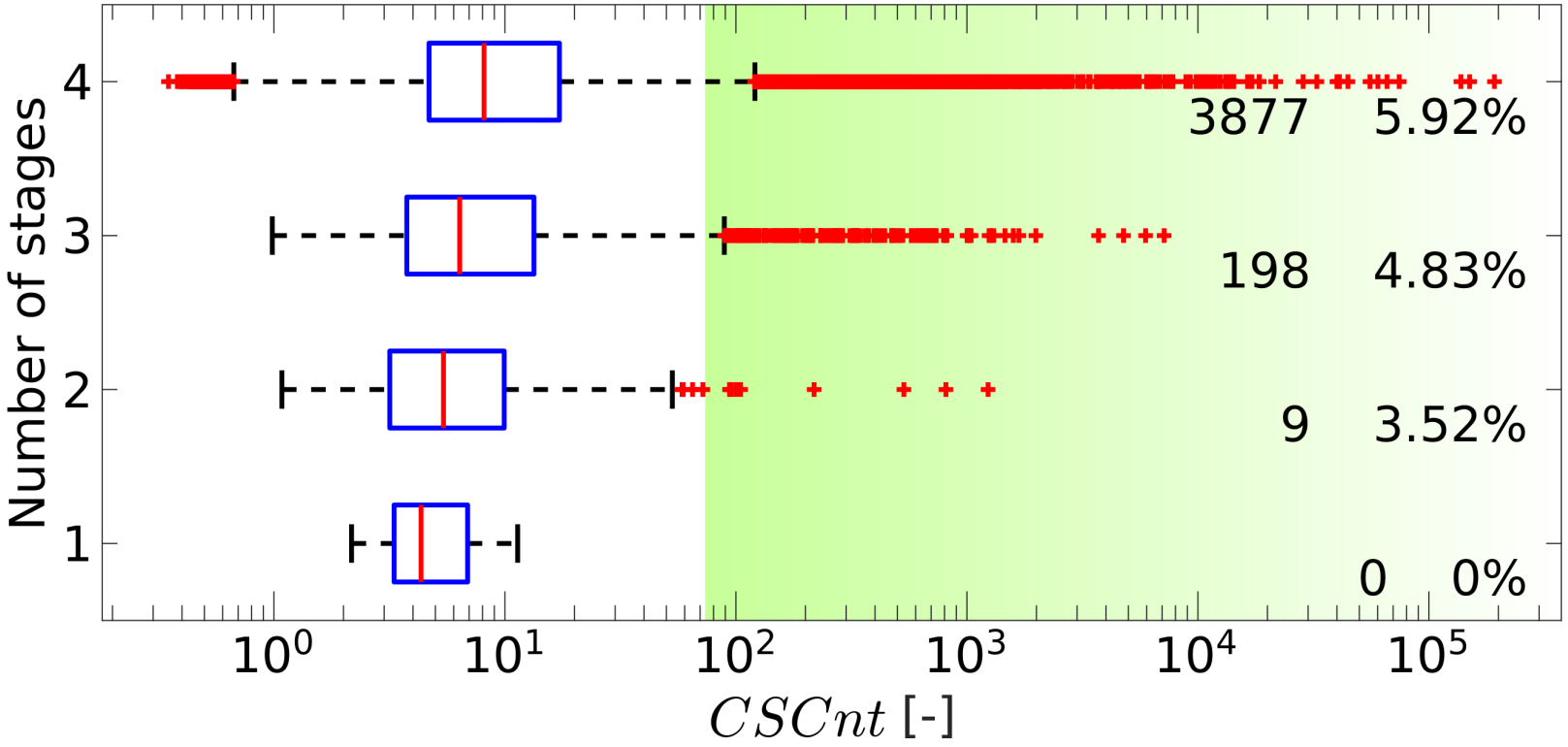
- SmartFusion2 results:



Configurable RO length

How many stages are necessary?

- Spartan 6 results:



Results random bit generation

	Spartan 6	SmartFusion2
Min $\mathrm{E}[CSCnt]$	74	103
Obtained $\mathrm{E}[CSCnt]$	81.12	107.85
Throughput [Mbit/s]	3.30	1.47
Min-entropy [bit/bit]	0.95	0.93
Area [DFF/LUT]	39/108	38/111

Comparison with other work

Architecture	FPGA family	Area [DFFs/LUTs]	Throughput [Mbit/s]	Statistical test	Design effort
This work	**Spartan 6**	**39/108**	**3.30**	**AIS-31 T6-T8**	-
	SmartFusion2	**38/111**	**1.47**	**AIS-31 T6-T8**	-
Original COSO [15]	*Spartan 6*	3/18	0.54	AIS-31 T8	MP
	Cyclone V	3/13	1.44	AIS-31 T8	MP
	SmartFusion2	3/23	0.328	AIS-31 T8	MP
COSO: one bit per half cycle [17]	*Actel Fusion AFS600*	7/24	2	NIST SP 800-22	MP & MR
	Spartan 3	7/18	1.6	NIST SP 800-22	MP & MR
COSO: mutual sampling [17]	*Actel Fusion AFS600*	14/29	4	FIPS 140-2	MP & MR
	Spartan 3	14/23	3.2	FIPS 140-2	MP & MR
COSO: parameter adjustment [14]	*Virtex-5*	109 slices	4.08	NIST SP 800-22	MP & MR
DC-TRNG [18]	*Spartan 6*	128 slices	1.1	AIS-31 T6-T8	MP
	Cyclone V	273 ALMs	1.116	AIS-31 T6-T8	MP & MR
PLL-TRNG [18]	*Spartan 6*	190 slices	1.0416	AIS-31 T6-T8	PLL required
	Cyclone V	273 ALMs	1.04	AIS-31 T6-T8	PLL required
ES-TRNG [11]	*Spartan 6*	5/10	1.15	AIS-31 T0-T5	MP
	Cyclone V	6/10	1.067	AIS-31 T0-T5	MP
TERO [15]	*Spartan 6*	12/39	0.625	AIS-31 T8	MP & MR
	Cyclone V	12/46	1	AIS-31 T8	MP & MR
	SmartFusion2	12/46	1	AIS-31 T8	MP & MR
STR [15]	*Spartan 6*	256/346	154	AIS-31 T8	MP & MR
	Cyclone V	256/352	245	AIS-31 T8	MP & MR
	SmartFusion2	256/350	188	AIS-31 T8	MP & MR

Thank you for your attention