# D-Type Flip Flop Entropy Conditioning

Sam Green
CS293G

June 15, 2017

## 1 Introduction

A common method of realizing a true random number generator (TRNG) for cryptographic applications is to sample an oscillator, or clock, exhibiting an unstable center frequency by an oscillator typically exhibiting a relatively stable center frequency. In the context of random number generation, the unstable oscillator is the source of entropy and will be referred to as $clk_{rand}$; the stable oscillator will be referred to as $clk_{ref}$.

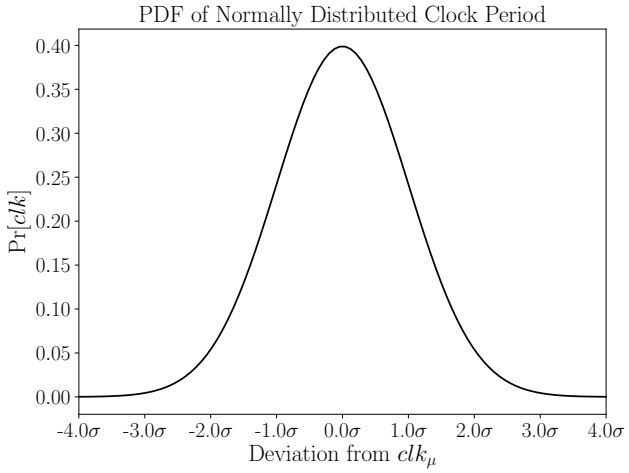


Figure 1: Normal period distribution

Initially, let us assume that the periods composing $clk_{rand}$ will be normally distributed, see Fig. 1. When $clk_{rand}$ is normally distributed, with equal length high (1) and low (0) phases of each period, knowledge of its clock period length is sufficient for determining how to generate entropy from it; that is knowledge of $clk_{rand}$'s period allows us to calculate how often to sample from it. There are

various means of generating entropy from an unstable clock. A common method is to sample $clk_{rand}$ at a sampling rate much lower than the center frequency of $clk_{rand}$. At any given time an oscillator can be assumed to be either high (1) or low (0), that is, ignore transition phase of clock; if the oscillator can be sampled at values outside of the 0/1 range, e.g. sampled values are continuous in the range [0,1], then we can normalize and round to fit the assumed range. Let $x \sim clk_{rand} \in \{0, 1\}$ denote the sampled state of the unstable oscillator.

When using the random sampling technique described above, the high and low phases of $clk_{rand}$ must sum to equal values in order for there to be no bias in the sampled numbers. As a first-order example, Fig. 2 shows three (unstable) clock periods, where the high and low half-periods $p_i$ are equal. When using the random sampling method, as long as $\sum p_i^{lower} / \sum p_i^{upper} = 1$, then unbiased numbers will be produced. Note that the previous summation does not require that all clock periods have the equal half-periods, as long as the sum of the upper half periods equals that of the lower. Intuitively, if such an equality exists, then randomly sampling $clk_{rand}$ (at the proper interval) will result in 0 or 1 with equal probability.

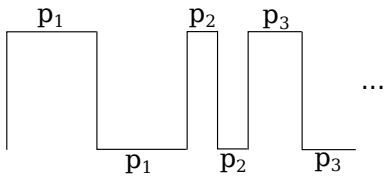


Figure 2: Unstable clock with equal half-periods

From this point forward, period will strictly be used to describe square waves with equal length high and low phases. Indeed, if a square wave has unequal length of high and low periods, then the underlying wave is actually composed of multiple sin waves, i.e. the typical $f = 1/T$ formula does not apply. Of course any physical instance of the stable oscillator $clk_{ref}$ will exhibit drift in period length, e.g. Fig. 1. As an aside, because frequency drift, even $clk_{ref}$ can be sampled for a source of entropy, as long as each square wave fits our strict requirement of a period.

Unintuitively, the distribution of periods exhibited by $clk_{rand}$ don't have to be normal in order to generate randomness. Suppose $clk_{rand}$ exhibited an skewed distribution of periods as shown in Fig. 3. As long as the *equality condition* $\sum p_i^{lower} / \sum p_i^{upper} = 1$ holds, then the random sampling method discussed previously will generate high entropy bits. In fact, any distribution of periods is adequate as long at the equality condition is met.

We will now turn our attention to the case where the sum of periods generated by a clock are not guaranteed to satisfy the equality condition. In this case, we can no longer sample the oscillator and directly use the resulting value. Instead, there will be 0/1 bias present that must be removed by an entropy conditioner. This situation can arise in practical TRNG applications where physical entropy sources are typically biased or gets biased as a result of ag-

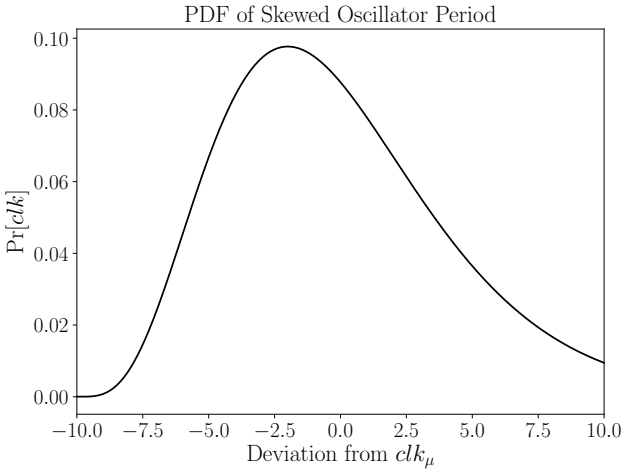


Figure 3: Skewed period distribution

ing and environmental effects. Thus, the raw output bits from these entropy sources will have some level of bias that must be removed. Even without bias, it is important to have a post-processor for additional security and reliability. The following section presents a means of achieving a low bias TRNG using minimal and standard CMOS or FPGA resources.

# 2 D-Type Flip Flop Entropy Conditioning

As stated in the introduction, a common method of building a TRNG for cryptographic applications is to occasionally sample $clk_{rand}$ with $clk_{ref}$. What "occasionally" actually means is dependent on the stability and frequency of $clk_{rand}$ with respect to $clk_{ref}$, but typically it is desirable to allow $clk_{rand}$ to run $\gg clk_{ref}$. If $clk_{rand}$ *exactly* satisfies the equality condition, then it is sufficient to directly use the sampled bits as the output of the TRNG, and it will have perfect entropy. If $clk_{rand}$ does not satisfy the equality condition, there will be a 0/1 bias in the sampled distribution. An alternative to directly sampling $clk_{rand}$ is to calculate or measure the mean period of $clk_{rand}$, $\mu_{rand}$, and then use $\mu_{rand}$ as the 0/1 reference point. How to accomplish this follows.

Assume the period of $clk_{rand}$ has some fixed distribution and that the underlying low and high periods may or may not satisfy the equality condition. We can initialize $clk_{rand}$ to high (1), then activate it. $clk_{rand}$ will stay at 1 for some time, then transition to low (0), and finally transition back to 1. The time between activation and the (moment before) the return to 1 is the length of the period. In any physical system $\mu_{rand}$ will have jitter or temporal noise, therefore, if $clk_{rand}$ were allowed to run freely, $clk_{rand}$ would complete a period

every $\mu_{rand}+\epsilon$ seconds, where $\epsilon$ may be considered i.i.d. (We can argue that the intrinsic thermal noise in circuit components translates into timing noise that is where the IID comes from.IC.)

At activation time, we may also reset $clk_{ref}$ and use it to measure the time it took $clk_{rand}$ to cycle. At the end of time $\mu_{rand}$, we sample $clk_{rand}$. If $clk_{rand}$ is still 0 at time $\mu_{rand}$, output random bit is set as 0; if $clk_{rand}$ has transitioned to 1, then it is set as 1. Whether or not $clk_{rand}$ has transitioned at time $\mu_{rand}$ is entirely dependant on $\epsilon$. For notational convenience, from here on, we will consider $clk_{rand}$ to be a random variable with $\epsilon$ being implicit.

If the mean period of $clk_{rand}$ is normally distributed, then it would have a probability density function as shown in Fig. 1. If this is the case, it is sufficient to sample $clk_{rand}$ every $\mu_{rand}$ seconds and directly use the sampled values as the TRNG output.

If $\mu_{rand}$ is not normally distributed, then sampling at $\mu_{rand}$ (or any other time) will lead to a bias toward 0 or 1 depending on the distribution. It is still cryptographically secure to use these biased numbers, if the bias is known, and if the bias can be removed via post-processing (aka conditioning or whitening). There are a number of common techniques used for entropy post-processing:

- von Neumann – consider two biased TRNG bits at a time. If the two bits are equal, then they are discarded. If the bits equal "10", then output 1. If the bits equal "01", then output 0.
- Hash function – The biased bits can be processed through a hash function, e.g. SHA-3.
- PRNG – The biased bits can initialize (seed) a pseudo-random number generator (PRNG). The PRNG will then output unbiased bits.

There are other similar methods. The von Neumann whitener removes all bias, but it discards many bits. The hash function and PRNG techniques will provide unbiased output, but if an attacker knows the underlying algorithm, which should be assumed under Kerckhoff's principle, the overall entropy reduces to the biased entropy of the inputs.

We present a new method which utilizes the central limit theorem (CLT). The CLT states that when independent random variables are added, their sum tends toward a normal distribution, whether or not the original variables are normally distributed. Initially, let

$$S_n := \frac{X_1 + ... + X_n}{n}. \tag{1}$$

The Lindeberg-Lévy CLT states the following: If $\{X_1, X_2, ...\}$ is a sequence of i.i.d. random variables with $\mathrm{E}[X_i] = \mu$ and $\mathrm{Var}[X_i] = \sigma^2 < \infty$, then as $n$ approaches infinity, the random variables $\sqrt{n}(S_n - \mu)$ converges to normal $N(0, \sigma^2)$:

$$\sqrt{n}\Big(S_n - \mu\Big) \to N(0, \sigma^2). \tag{2}$$

By utilizing the CLT, along with readily available hardware resources, we present an efficient design to remove bias from $clk_{rand}$. We achieve this by performing the operations of Eqs. 1 and 2 via statistical properties implicit in our design. Our entropy conditioner uses a chain of $N + 1$ D-type Flip Flips (DFFs). DFFs were chosen because they are a readily available resource in FPGAs and they are simple to implement in custom silicon. The first $N$ DFFs will cycle through 0's and 1's alternately, and they are clocked by $clk_{rand}$, as depicted in Fig. 4. The last DFF is the final sampling register driven by $clk_{ref}$.

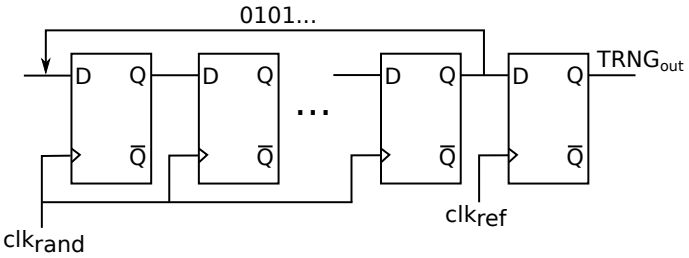


Figure 4: DFF chain driven by ustable clock $clk_{rand}$ and sampled by stable and accurate clock $clk_{ref}$.

At the beginning of random number generation, we initialize the size-$N$ DFF chain with all 0's and reset a timer, using the stable reference clock $clk_{ref}$. We then place a 1 at the input of the first element of the chain and activate $clk_{rand}$ and $clk_{ref}$. $clk_{ref}$ is used as a reliable counter to sample the output of the DFF chain at time $T = N \times \mu_{rand}$.

The first DFF output will transition from 0 to 1 at time $clk_{rand1} = \mu_{rand} + \epsilon_1$, where $\mu_{rand}$ is measured in the number of cycles of $clk_{ref}$ necessary to span the time $\mu_{rand}$. The second DFF will then transition from 0 to 1 at time $clk_{rand1} + clk_{rand2} = 2 \times \mu_{rand} + \epsilon_1 + \epsilon_2$. The $N^{\text{th}}$ DFF will transition at time

$$N \times clk_{rand} + \sum_{i=1}^{N} \epsilon_i. \tag{3}$$

Because of the feedback shown in Fig. 4, the $N^{\text{th}}$ DFF will toggle between 0 and 1 every $N^{\text{th}}$ cycle of $clk_{rand}$. Note that Eq. 3 is similar to Eq. 1, and therefore Eq. 2 applies; that is, the sampled output of the $N$ DFFs will approach a normal distribution with a mean of $\mu_{rand}$. When $clk_{rand}$ is biased, the DFF period will not be exactly normal until $N$ approaches infinity. However, even under practical resource constraints, for example using $N = 10$, there is still benefit to this approach and bias is significantly reduced, as will be demonstrated in the next section.

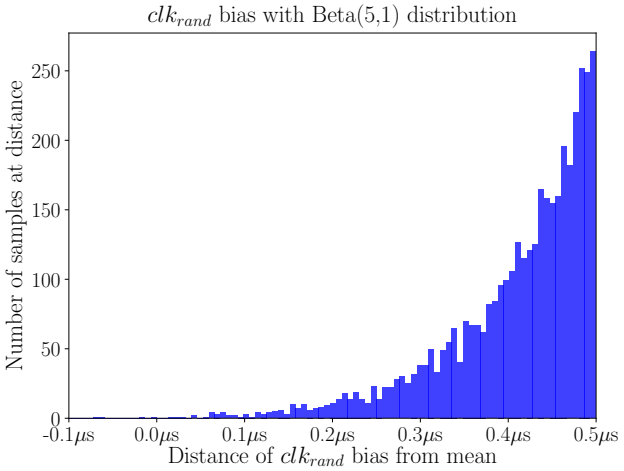


Figure 5: Histogram of bias values ($\epsilon$'s) from 4,000 simulated $clk_{rand}$ periods. The biases are distributed Beta(6,1) and then centered at 0$\mu$s.

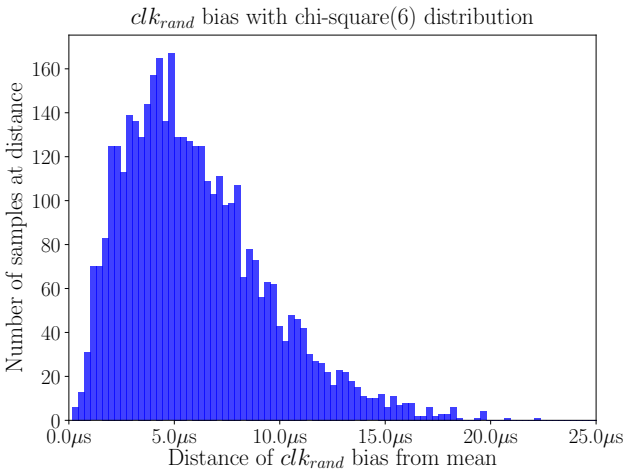


Figure 6: Histogram of bias values ($\epsilon$'s) from 4,000 simulated $clk_{rand}$ periods. The biases are sampled from chi-square distribution. This clock would result in biased numbers if sampled directly.

# 3 Simulation Results

For this simulation, three $clk_{rand}$s were simulated: one with no bias, i.e. normally distributed, and therefore ideal, one with biases from a Beta(6,1) distri-

bution, and one with biases from a chi-square(3) distribution. The histograms of 4,000 samples from the two biased clocks are given in Figs. 5 and 6.

Directly sampling the biased $clk_{rand}$s would result in a skewed distribution of 1's and 0's. Skew in the 0/1 distribution results in less available entropy. *Entropy* is a population descriptor similar to population mean and variance; in practice, Partial Average Shannon Entropy (ASE) is used for cryptographic assessment; this metric is similar to using sample statistical metrics. ASE is formally defined as:

$$ASE = -\frac{1}{k}\sum_{\beta=0}^{2^k-1} P(\beta)\cdot log_2 P(\beta), \tag{4}$$

where

- ASE is expressed in bit/time-step of the underlying TRNG.
- $k$ is a window size to partition the TRNG output for the ASE calculation.
- $\beta$ takes on all possible $2^k$ integer values from a $k$-bit window, i.e. $\beta = 0 \ldots 2^k - 1$.
- $P(\beta)$ is the generation probability of $\beta$.

In other words, the ASE returns average entropy per bit over a sequence of $M$ bits, parsed into $k$ bit words. In cryptographic applications, the closer the calculated ASE is to 1, the better the underlying entropy. The mathematical proof is beyond the scope of this paper, but, given a fixed $M$, the larger $k$ is, the better the underlying entropy must be in order to maintain proximity to 1. Therefore an insightful entropy analysis heuristic is to take a fixed sequence of $M$ bits from a TRNG and calculate the ASE over increasing values of $k$.

We used the ASE metric to evaluate our DFF-based entropy conditioner. We first generated 4,000 random bits from the ideal source, the two biased sources, and the output of the $N = 10$ DFF-based entropy conditioned sources. Figs. 7 and 8 compare the raw/biased outputs from the two biased sources to the ideal source and the DFF-corrected source. It can be observed in Fig. 7 that the DFF-conditioned output significantly reduces bias and brings the measured ASE close to the ASE of the ideal source. The chi-square source is already close to normal, so the bias and, therefore correction, are not as extreme, as can be seen in Fig. 8. In both cases the DFF-corrected bits have ASE very close to ideal for each value of $k$; this shows that even with 10 DFFs, we are able to significantly improve a TRNG. Also note that conditioning an ideal normal source with our DFF-based entropy conditioner will have no effect on the output ASE – that is if ASE equals 1 before conditioning, then ASE will equal 1 after conditioning – and may therefore be applied in all cases.

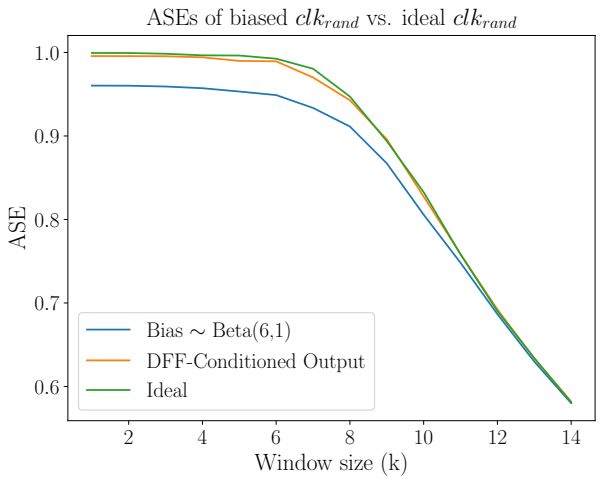


Figure 7: Results when applying DFF-based entropy conditioner to Beta(6,1)-biased entropy source. 10 DFFs used for conditioning. Conditioned results approach ideal (normally distributed) after conditioning.

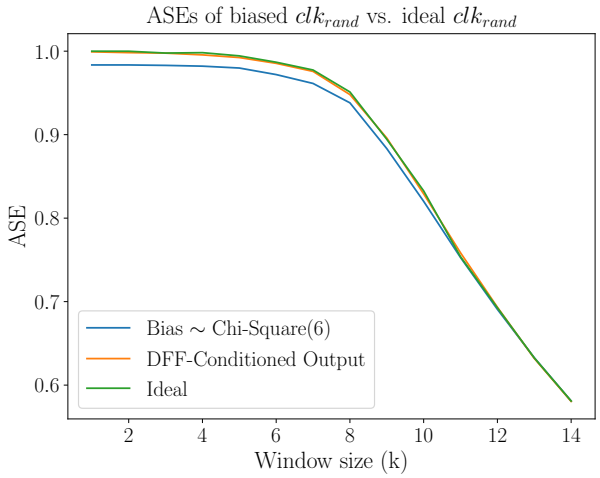


Figure 8: Results when applying DFF-based entropy conditioner to chi-square(3)-biased entropy source. 10 DFFs used for conditioning. Conditioned results approach ideal (normally distributed) after conditioning.