Approximate Computing for Privacy in IoT devices

Kamran Hasan
University of Toronto
kamran.hasan@mail.utoronto.ca

Karthik Ganesan
University of Toronto
karthik.ganesan@mail.utoronto.ca

Natalie Enright Jerger
University of Toronto
enright@ece.utoronto.ca

Abstract
Privacy is an increasingly essential requirement for modern computing systems. Privacy requires true random numbers often generated using dedicated hardware. We show how approximate computing can be used for generating random numbers. We demonstrate a proof of concept using a system with an approximate adder. We show that there is a fundamental synergy between approximate computing and random number generation, allowing other approximate computing techniques to also be used for random number generation on resource constrained devices.

1 Introduction
Security and privacy are critical requirements for modern day computing systems. Cryptographic algorithms, used to provide security and privacy, rely on a key or a one-time nonce [11, 24, 25, 27, 28] to secure information. Such keys and one-time nonces need to be randomly generated to prevent key recovery through cryptanalysis. Random numbers produced in digital systems are typically pseudo-random as they follow a predictable pattern (e.g., \texttt{rand}) in C++. `$Random in unix). Cryptographically secure true random number generation (TRNG) requires dedicated hardware. IoT devices require TRNG to encrypt sensitive information they collect (e.g., users biometric data) [7]. State-of-the-art low-power TRNG hardware can occupy up to 6000 $\mu m^2$ [32]. As a low-power IoT processor such as the ARM M0+ is only 71,000 $\mu m^2$ [19], TRNG hardware can take up 8.5% of the chip area. To reduce this overhead, we show that existing support for approximate computing can be further leveraged for TRNG. We demonstrate our approach using a system adapted for near-threshold computing. We begin by providing some background on near-threshold computing as well as voltage regulation in IoT devices, which we leverage for our technique.

2 Background
Near threshold computing. Digital circuits typically operate at voltages greater than the threshold voltage ($V_{th}$) of the transistors used. While this allows for operating at higher frequencies, it comes at the cost of increased energy [18]. A lower supply voltage saves energy up until the voltage reaches $V_{th}$, after which energy once again increases due to sub-threshold leakage [18]. Thus, the voltage that minimizes energy is just above the circuit’s $V_{th}$. Operating close to this voltage is known as near threshold computing (NTC) [12].

NTC imposes an increase in circuit delay and therefore a reduction in clock speed [12]. For example, a RISC-V CPU for IoT applications operates at 357 MHz at 1.08V and 50MHz at an NTC voltage of 0.6V [14]. Despite this, NTC is applicable for IoT devices as they are typically used to sense physical phenomenon (e.g., temperature, heart rate) which change at Hz or kHz speeds [20]. To this end, prior work has used NTC to develop ultra-low power SoCs for edge computing [21, 22].

Operating near $V_{th}$ also makes NTC circuits more prone to errors caused by process variation [17]. Building on this, prior work has used NTC circuits to build approximate logic [17, 23]. While the CPU datapath can use NTC circuits, many parts of the chip cannot tolerate any errors (e.g., CPU control path). Therefore NTC circuits must have two voltage rails, one much higher than $V_{th}$ and one near-$V_{th}$ to support both precise and approximate logic. We take advantage of these two voltage rails to generate random numbers. To elaborate, we present some background on power regulation in low-power IoT devices.

Voltage regulation. Power regulation is ubiquitous in electronic devices and even a low-power IoT device uses some form of power conversion, with the most common being switch mode power supplies (SMPS) [1]. SMPS work by rapidly switching two field-effect transistors (FETs) on and off to maintain the desired output voltage. This is controlled by a periodic switching signal from a gate driver chip. The effect of this switching is that the output of the SMPS is not a pure DC signal but instead is composed of a low frequency (LF) AC ripple as well as a high frequency (HF) noise component (Figure 1a). The frequency of this HF noise is the switching frequency of the gate driver. While the HF noise is often seen as undesirable, our technique uses this noise inherent to ubiquitous SMPS circuitry to enable TRNG.

3 TRNG with Near Threshold Computing
Our technique builds on existing circuitry needed to enable NTC. Specifically, NTC requires two voltage rails: a regular voltage $V_{nom}$ and a near-threshold voltage $V_{droop}$. For low cost, low power IoT devices, having two voltage regulators would be cost and area prohibitive, so we use a simple resistor divider to generate the two rails. This resistive network would also transmit the HF noise inherent to SMPS, which becomes the entropy source [15] necessary for TRNG. This HF noise is passed through some digital logic to generate random bits.
While our technique works with any digital circuit, we use adders due to their ubiquity. Our baseline system supports two adder modes: approximate and precise. Similar to prior work, the programmer annotates sections of code that can be approximated [26]. To maximize energy savings, our design defaults to running in approximate adder mode (AAM) whenever precise addition is not explicitly needed. When the system is in AAM, the enable_AAM signal to the adder is asserted (Figure 1b). This switches the $V_{dd}$ of the adder from $V_{nom}$ to $V_{droop}$. Since the transistors in the adder are now operating close to $V_{th}$, the HF noise will force them to randomly switch on or off. Since we can only generate random bits during the HF noise period, we use the gate driver switching signal from the regulator ANDed with the enable_AAM signal to switch the adder inputs $A$, $B$ and $C_{in}$ to ‘1’, ‘1’ and ‘0’. This causes the adder to produce random bits, which we feed to a Schmitt trigger (which serves as a 1-bit analog-to-digital convertor). These random bits are stored in a register until needed. For clarity, we only show 1-bit of the adder in Figure 1b, however our design can scale to multiple adder bits.

We use multiplexers and de-multiplexers in Figure 1b for clarity. In reality, our system realizes this functionality using much cheaper FETs. Thus, our solution costs just 8 FETs per adder bit and a single AND gate.

4 Evaluation

We use Simulink [2, 5, 6, 10, 13], to simulate the circuit shown in Figure 1b. We use our design to generate 1024 bits in Simulink; this output pass all tests in the NIST 800-22 test suite [8]. The NIST test suite is widely used for verifying that random numbers are sufficiently uncorrelated for cryptographic purposes.

Case study. We demonstrate the efficacy of our technique for biomedical applications. Biomedical devices, such as wearable and implantable devices for health monitoring, are an increasingly important class of IoT devices. Our technique is well suited to these biomedical applications for the following reasons:

- Wearable and implantable devices are extremely energy constrained, requiring the use of stringent energy conservation methods.
- Biomedical applications are error tolerant, making them well suited for approximate computing techniques [30].
- Data privacy is of paramount importance for biometric information [9].
- Biological signals are sampled at frequencies of a few Hz, requiring only modest rates of TRNG [20].

Based on these criteria, we show that the rate of TRNG using our technique is sufficient for this important class of IoT devices. The rate of TRNG in our system is calculated as:

$$
\text{Bits/sec} = \nu \cdot \gamma \cdot f_{sw}
$$

where:
- $\nu$ is the number of adder bits approximated.
- $\gamma$ is the fraction of time the system spends in AAM.
- $f_{sw}$ is the switching frequency of the SMPS.

Prior work has shown a <1% output error by approximating up to 8-bits of an NTC adder used in an FIR filter [29]. Thus, we set $\nu=8$, to approximate the lower 8-bits of the adder. As the time spent in AAM is highly application and architecture specific, we assume a nominal value of $\gamma=0.5$. The rate of the HF noise varies greatly across different SMPS devices. We use a range of values for $f_{sw}$ from 100KHz to 2.4MHz [3]. Solving equation 1 yields a range TRNG rates from 390 Kbps to 9.15 Mbps. Biomedical applications typically transmit at much lower rates (e.g., 250 Kbps for retinal prosthetics, 10 Kbps for pacemakers and 300 Kbps for cochlear implants [31]). Thus, the rate of TRNG of our technique is sufficient to encrypt data being transmitted from these devices.

5 Conclusion

We show how existing support in the form of multiple voltage rails for near-threshold computing can be further leveraged for TRNG. We hope to motivate the exploration of such synergy in other approximate computing work, particularly in the constrained environments of IoT devices.

---

1We assume a 32-bit adder in our design.
References


