2023 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE 10 YEAR ANNIVERSARY

System-Level Simulation of a SPAD-Based Time-of-Flight Sensor in SystemVerilog

Seungah Park[1], Hyeongseok Seo[2], Canxing Piao[2], Jaemin Park[3], Jaehyuk Choi[1,2], Jung-Hoon Chun[1,2]

[1]Department of Electrical and Computer Engineering, Sungkyunkwan University, Suwon, Korea

[2]SolidVue Inc., Seongnam, Korea

[3]Scientific Analog Inc., Seoul, Korea

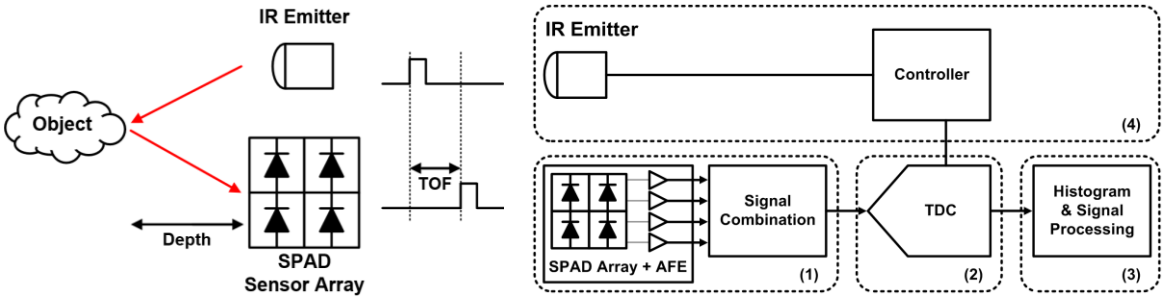
psa8789@skku.edu, hsseo@solid-vue.com, pcanxing@solid-vue.com, jaemin@scianalog.com, choix215@skku.edu, jhchun@skku.edu

***Abstract*—This paper introduces a system-level simulation platform developed in SystemVerilog for a SPAD (Single Photon Avalanche Diode)-based direct Time-of-Flight Sensor. The distinct characteristics of SPAD, including random noise and photon detection attributes, are statistically modeled using the Poisson process. Utilizing the system-level verification platform, our proposed SPAD model ensures a comprehensive assessment of depth sensor systems, covering various incident light conditions and SPAD characteristics. It is demonstrated that a 1.5-meter range proximity sensor, equipped with 64 SPADs, a multi-event folded-flash TDC, a 12-bit histogram counter etc., completes its simulation in just 4.6 minutes, yielding 4095 histogram data points.**

Keywords—Single-photon avalanche diode (SPAD); Time-of-flight sensor; Modeling; SystemVerilog

I. INTRODUCTION

The Direct time-of-flight (TOF) sensor is a device that provides distance (Depth) information by measuring the time difference between a light pulse emitted from the sensor's emitter and the light pulse incident on the sensor after being reflected from an object, as illustrated in Fig. 1(a). The sensor's receiver utilizes a single-photon avalanche diode (SPAD), which exhibits high sensitivity to light [1]. Direct TOF sensors, such as light detection and ranging (LiDAR) sensors, consist of four primary components [2], as depicted in Fig. 1(b): a SPAD array and analog front-end (AFE) circuits followed by a signal combiner, time-to-digital converter (TDC) that converts the photon detection time to digital data, histogramming and digital signal processing unit, and timing controller for the emitter and receiver circuits.



(a) (b)

Figure 1. (a) Concept of direct time-of-flight measurement and (b) a simplified diagram of a SPAD-based sensor.

To predict the accuracy of the TOF sensor under development, it is crucial to create a model that simulates the physical and statistical operating characteristics of the SPAD. Furthermore, a robust simulation platform that focuses on the integration of the SPAD model with the entire sensor system is essential. In this work, a SPAD-based sensor system verification platform is implemented using SystemVerilog and applied to a proximity sensor with a relatively simple configuration to assess the verification accuracy. XMODEL primitives [3] are utilized in the modeling of SPAD and the proximity sensor to express precise timing information without being limited by the

timestep of SystemVerilog simulation. The proximity sensor is designed to measure distances of up to 150 cm and achieve a resolution time of 0.2 ns. This system-level simulation will be employed in the design and verification of SPAD-based sensors, such as mid- to long-range LiDAR systems with high complexity, in the future.

II. Statistical Behavioral Modeling of SPAD

Fig. 2 illustrates the operation of the SPAD and AFE circuit to be modeled. The SPAD functions in Geiger mode, with a reverse bias higher than the breakdown voltage (V_{BD}) applied. Upon the incidence of a photon on the SPAD, an avalanche current is generated, and as this current passes through the AFE, a digital pulse is generated. The width of the digital pulse is referred to as the SPAD dead-time, Tdead, because during this dead-time, another digital pulse will not be generated even if a new photon arrives at the SPAD. The SPAD also responds to randomly generated dark current. The frequency of the noise pulses due to dark current is quantified as dark count rate (DCR). For simulation of SPAD-based sensors, physics-based SPAD models [4, 5] that emulate the detailed dynamics of SPAD's behavior can be used. However, it is essential to implement a model that can effectively represent the stochastic distribution of the SPAD's response time to incident photons and random noise, and incorporate this model into the overall system-level simulation. We propose a simple, yet effective SPAD model that takes DCR as an input variable, generates output pulses through a Poisson process, and determines the timing of photon-induced pulse generation based on the incident light intensity and the photon detection probability of the SPAD.

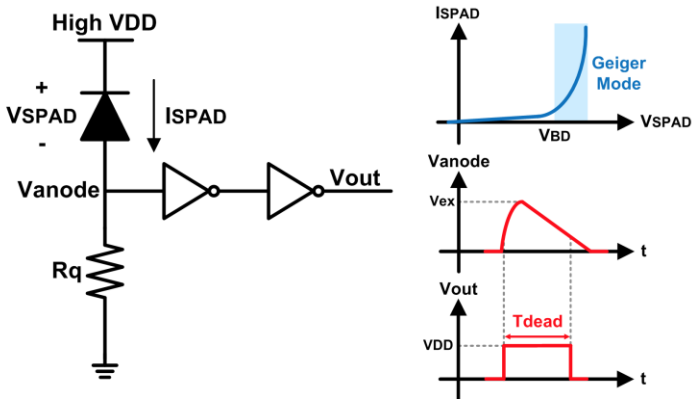


Figure 2. The operation of SPAD and analog front-end circuit.

Fig. 3 illustrates the proposed SPAD model, with detailed descriptions for the inputs in Table I. In the "Dark Count Generator", the "Tnoise Generator" generates the randomized timing of noise pulse initiation, *Tnoise*. The mean frequency of noise pulse generation is determined by the input parameter *DCR*. To generate pulses at the time of *Tnoise*, the model utilizes an integrator and slicer, and the subsequent "Single Pulse Generator" creates output pulses, *noise_pulse*, with a user-defined pulse width of *Tdead*. The "Photon Detector" is similar to the "Dark Count Generator," but it takes the light pulse shape (*pulse_shape*), photon arrival time (*TOF*), light intensity (*INT*), and photon detection probability (*PDP*) as inputs, and uses these values to calculate the photon-induced pulse generation time (*Tph*). Finally, the "Dead time Controller" compares *ph_pulse* with *noise_pulse* and and if there is an overlap between the two pulses, only the first generated pulse is extracted as the *SPAD_pulse*. The proposed modeling can be easily customized by adjusting the input values to match the specifications of the designed SPAD and the measurement environment.

Table I. The input information.

Input	Description	Unit
RSTB	Reset signal for operation	
Tdead	Dead-time of SPAD	ns
pulse_shape	Light pulse information (Resolution = 0.02 ns)	
TOF	Time-of-flight of the object	ns
INT	Light intensity	photons/pulse
PDP	Photon detection probability of SPAD	
DCR	Dark count rate	counts/ns

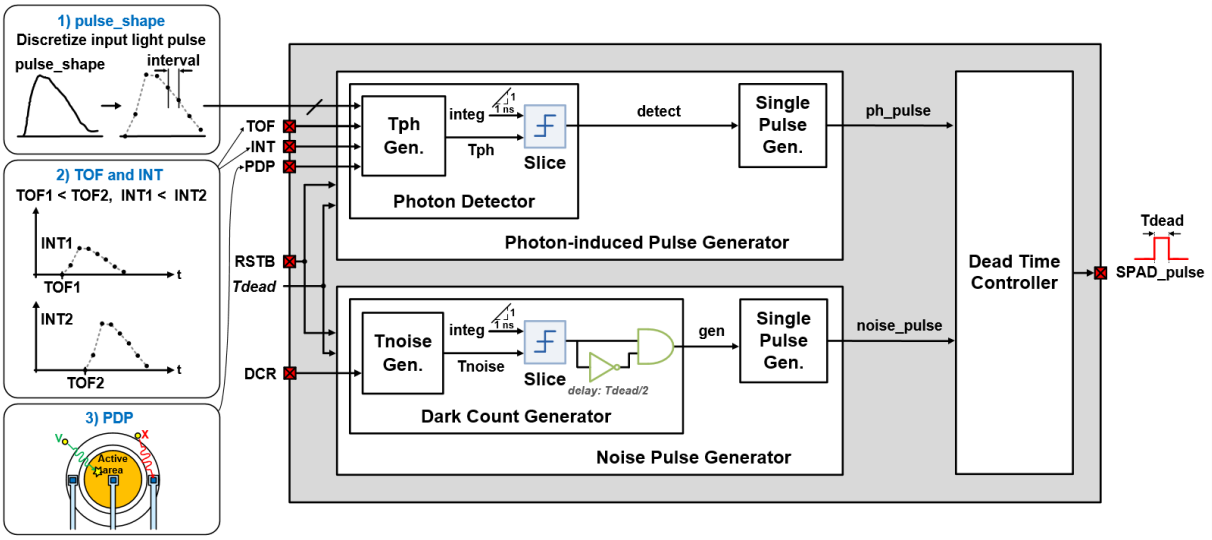


Figure 3. Block diagram of the SPAD model. The inputs for the Tph Generator are described on the left side.

A. *Tnoise Generator*

In the Tnoise Generator, the timestamp when a dark count takes place, denoted as *Tnoise*, is ascertained from the Poisson process. If we consider a Poisson process with a rate of λ, the probability that the first dark count generation time, X_1, exceeds time t can be expressed as:

$$P(X_1 > t) = P(no\ dark\ count\ generation\ between\ 0\ and\ t) = e^{-\lambda t} \quad (1)$$

Similarly, the probability that the first dark count generation occurs within the time frame t is given by:

$$P(X_1 \leq t) = 1 - e^{-\lambda t} \quad (2)$$

From (2), the interarrival time, t, can be deduced as:

$$t = \frac{\ln(1 - P(X_1 \leq t))}{-\lambda} \quad (3)$$

Fig. 4 presents the code of the Tnoise Generator, while Fig. 5 illustrates the timing diagram of the Dark Count Generator in Fig. 3. $P(X_1 \leq t)$ in (3) is replaced by the *rand_uniform(0,1)* function, which follows a uniform distribution over the interval [0,1]. Additionally, the rate λ is substituted for the input *DCR*. The generation time of the first noise pulse after *RSTB* transitions to a rising edge is determined using (3). When the linear integrator output, *integ*, intersects with the computed timing of *Tnoise*, *gen* in Fig. 3 generates a pulse with a width *Tdead/2*. *Tnoise* is continuously updated while *RSTB* is 1. If the updated *Tnoise* is less than *Tdead*, *Tnoise* is recalculated and added to the previous value until the sum exceeds *Tdead*. Also, since *Tnoise* is updated every time *gen* has a falling edge, *Tdead/2* - the width of *gen* - is subtracted from the *Tnoise* value determined by (3).

```
module Tnoise_gen
…
//For initial Tnoise (TDCR)
always @(posedge RSTB) begin
   //If random_dark = 1, ln(1-random_dark) = -inf (NOT possible)
   random_dark = rand_uniform(0,1);
   while(random_dark >= 1) random_dark = rand_uniform(0,1);
   TDCR = $ln(1-random_dark) / (-DCR);
end
```

```
//For updating Tnoise when gen has a falling edge
always @(negedge gen) begin
   TDCR = 0;  random_dark = rand_uniform(0,1);
   while(random_dark >= 1) random_dark = rand_uniform(0,1);
   TDCR = $ln(1-random_dark) / (-DCR);
   //For considering dead-time (Tdead)
   while(TDCR < Tdead) begin
      random_dark = rand_uniform(0,1);
      while(random_dark >= 1) random_dark = rand_uniform(0,1);
      TDCR = TDCR + $ln(1-random_dark) / (-DCR);
   end
   TDCR = TDCR – (Tdead/2);
end
assign Tnoise = TDCR;
endmodule
```

Figure 4. Tnoise Generator calculating the randomized timing of noise pulse initiation.

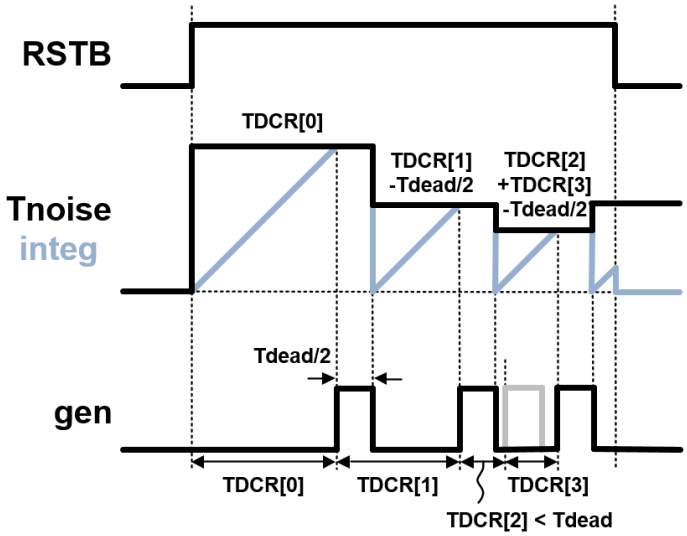


Figure 5. Timing diagram of the Dark Count Generator.

B. Tph Generator

Tph Generator calculates the timing of photon detection (*Tph*) based on the characteristics of the incident light, described by *pulse_shape* and its intensity (*INT*) through the lens of the Poisson process. The definitions and relationships of *pulse_shape*, *TOF*, *INT*, and *PDP* - inputs of Tph Generator - are illustrated on the left side of Fig. 3. Fig. 6 presents the code of Tph Generator. First, *pulse_shape_disc* is obtained by discretizing the input *pulse_shape*. Given that the *pulse_shape* used in the simulation has a fine precision of 0.02 ns, *pulse_shape_disc* adopts a resolution time (*Tres*) of 0.02 × *interval* ns. Second, *pulse_shape_int* is achieved by normalizing *pulse_shape_disc* and multiplying it by *INT*. Contrary to the Tnoise Generator, where the user-defined *DCR* input dictates the expected value of the Poisson process, in the Tph Generator, each datum of *pulse_shape_int* is taken as the expected value of the Poisson process. The next step employs the cumulative distribution function (CDF) of the Poisson process to calculate at which point within *pulse_shape_int* the SPAD reacts.

Let X represent a random variable conforming a Poisson process with a rate of λ. The probability of the SPAD responding n times is depicted as:

$$P\{X = n\} = \frac{e^{-\lambda}\lambda^{n}}{n!} \tag{4}$$

From (4), the CDF is derived as:

$$P\{X \leq n\} = \sum_{k=0}^{n} \frac{e^{-\lambda}\lambda^{k}}{k!} \tag{5}$$

Each datum from *pulse_shape_int* is inserted in place of λ in (5). The value of *k* increments by one until *cdf* surpasses *thres_cdf* which randomly derives its value from the *rand_uniform(0,1)* function. The derived *k* can be interpreted as the number of photons successfully reaching the SPAD for each expected value of *pulse_shape_int*. Then, given the SPAD's PDP, a determination is made as to whether each photon's arrival instigates an avalanche. In this context, for each photon's arrival, the randomly generated *thres_pdp* is compared with the *PDP* input value. For example, if the calculated *k* equals 2, the comparison between *thres_pdp* and *PDP* is performed twice. This process is sequentially conducted for each datum within *pulse_shape_int*. If the PDP criterion is met for the first time at the *i*-th datum, it means that a pulse was produced by a photon at the timestamp of *TOF* + *i* × *Tres* ns.

```
module Tph_gen
…
always @(posedge RSTB) begin
   signal_sequence = 0;    //Photon detection time within input light pulse, pulse_shape
   Tph = 0;                //Photon detection time (= TOF + signal_sequence)
   pulse_sum = 0;          //Sum of discretized pulse_shape data (pulse_shape_disc)

   //For discretizing pulse_shape (Resolution time of pulse_shape_disc = (interval * 0.02) ns)
   for(i=0; i<(pulse_len/interval); i++)    pulse_shape_disc[i+1] = pulse_shape[i*interval+int'(interval/2)+1];

   //For normalizing pulse_shape_disc
   for(j=0; j<(pulse_len/interval)+1; j++) begin
      if(pulse_shape_disc[j]<0) pulse_shape_disc[j] = 0;
      pulse_sum = pulse_sum + pulse_shape_disc[j];
   end
   for(k=0; k<(pulse_len/interval)+1; k++)   pulse_shape_int[k] = pulse_shape_disc[k] / pulse_sum * INT;
   Tres = 0.02 * interval;   //Tres: Resolution time of pulse_shape_int

   //For modeling photon arrival and SPAD avalanche
   for(i=0; i<(pulse_len/interval)+1; i++) begin
      k = 0;  cdf = 0;  kfactorial = 1;   thres_cdf = rand_uniform(0,1);
      while (1) begin
         cdf = cdf + $exp(-pulse_shape_int[i]) * $pow(pulse_shape_int[i], k) / kfactorial;
         if(cdf <= thres_cdf) begin
            //For considering PDP of the SPAD
            thres_pdp = rand_uniform(0,1);
            if(PDP > thres_pdp) begin
               signal_sequence = i * Tres;
               break;
            end
            k = k+1;  kfactorial = kfactorial * k;
         end
         else break;
      end
      if(signal_sequence != 0) break;
   end
   if(signal_sequence != 0) Tph = TOF + signal_sequence;
end
endmodule
```

Figure 6. Tph Generator calculating the photon-induced pulse generation time.

III. MODELING OF A SPAD-BASED SENSOR

Fig. 7 illustrates the overall block diagram of a SPAD-based sensor configured as a proximity sensor using the structure in [6, 7]. Since the *SPAD_pulse* from the SPAD model passes through the T flip-flop and XOR Tree, *TDC_IN* toggles whenever 64 SPADs react. The multi-event folded-flash TDC digitizes the timing of *TDC_IN* toggles, while the time resolution of TDC is determined by phase-locked loop (PLL) clock (*PLL_CLK*). Subsequently, a histogram is stacked to obtain an accurate TOF. The Histogram 12b Counter stacks the histogram using either the Ripple Counter (*SEL* = 0) or Shift Register (*SEL* = 1) method. By setting the number of counter bins to 12, up to 4095 measurements can be made. The histogram output, *Count_clk0~7*, can be plotted directly or post-processed by an FIR filter. The Controller module adjusts the overall system operation timing. To the best of our knowledge, a system-level simulation of the sensor, including SPAD modeling in SystemVerilog, has not been demonstrated.

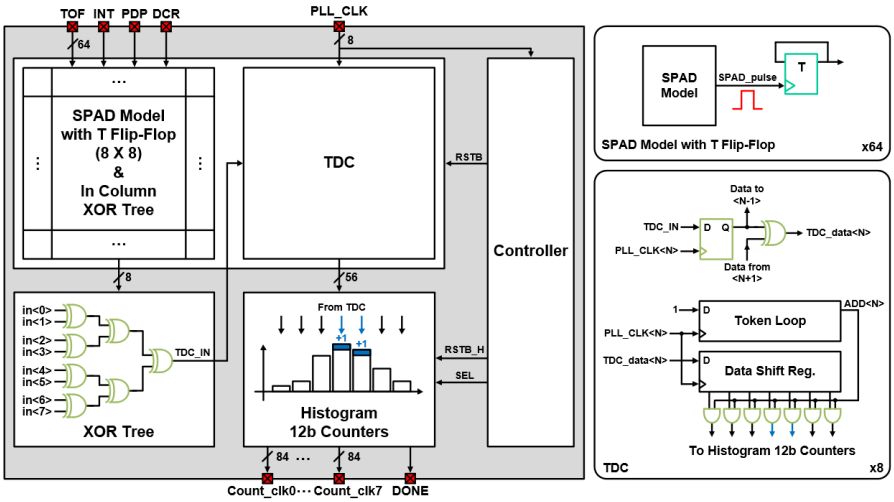


Figure 7. Overall Block diagram of the SPAD-Based Sensor model with details of each block.

IV. TESTBEHCH FOR A SPAD-BASED SENSOR SIMULATION

Fig. 8 illustrates a block diagram for the testbench of a SPAD-based sensor. The TB_HIST module consists of PLL Clock Generator, Sensor TOP, and Timing Aligner. In the PLL Clock Generator, the frequency of the 8-phase *PLL_CLK* was set to 600 MHz which determines the TDC resolution of 208 ps. The *PLL_CLK*'s RMS jitter was set to 6 ps using the *freq_to_clk* primitive cell. In the Sensor TOP module, the input values are defined through the "tb_hist.sv" file. Once the histogram has accumulated as many times as desired, the output *DONE* signal becomes 1, and the Timing Aligner rearrange the histogram outputs *Count_clk0~7* according to their actual timing, and the values are sequentially stored in the "histogram_output" file. Finally, the "plot.py" plots the values stored in the "histogram_output".

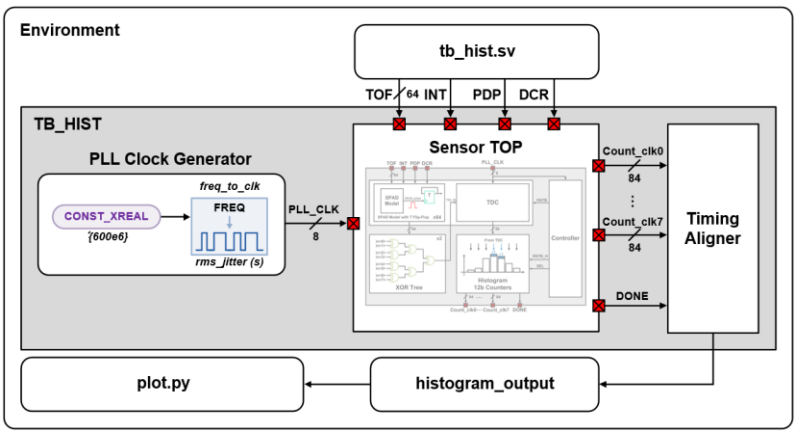


Figure 8. Block diagram of the testbench.

V. RESULTS

We performed a system-level simulation of the SPAD-based sensor using two different light pulses: one with a dispersed shape and the other with relatively sharp edges. The light pulse cycle was set to 15 ns, including a reset timing of 1.66 ns. It can measure up to a distance of about 153 cm, and a total convergence time of 65 us is required to measure TOF 4095 times. Fig. 9 shows the overall timing diagram of the simulated sensor operation. The total simulation time was about 4.6 minutes. Fig. 10 shows the histogram results for various input conditions, including changes in DCR, intensity, TOF, and PDP. It demonstrates that the histogram effectively tracks the trend of *pulse_shape*. We also successfully observed that the various design variables such as the number of histogramming, clock jitter characteristics and so on have an impact on the final TOF measurement results. We confirmed that the design of the sensor can be successfully verified with the proposed SPAD model and system-level verification platform.

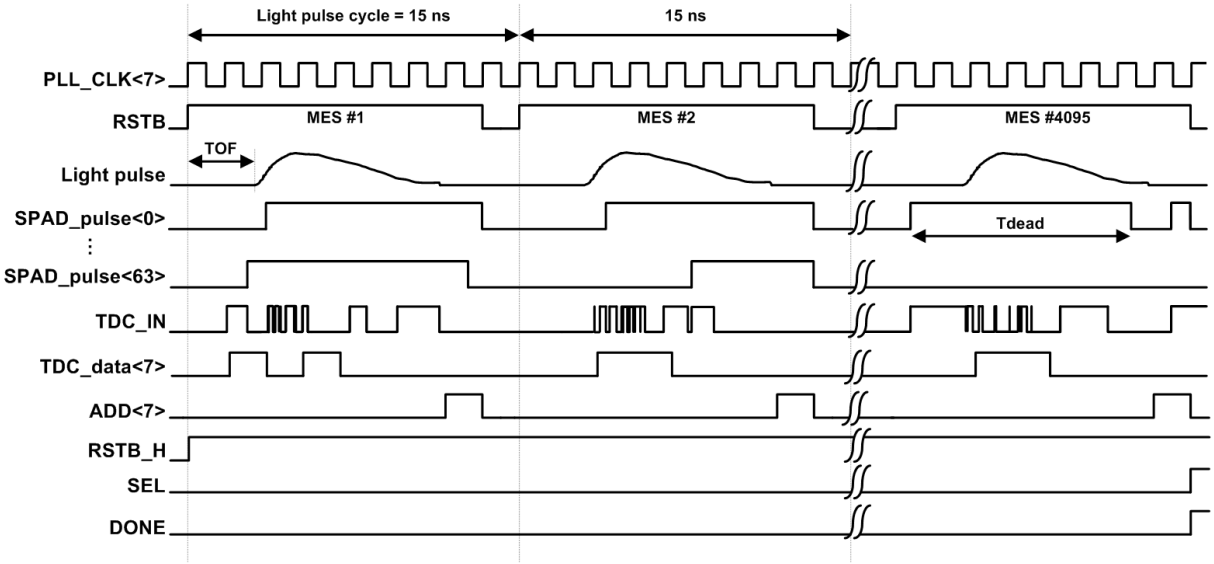
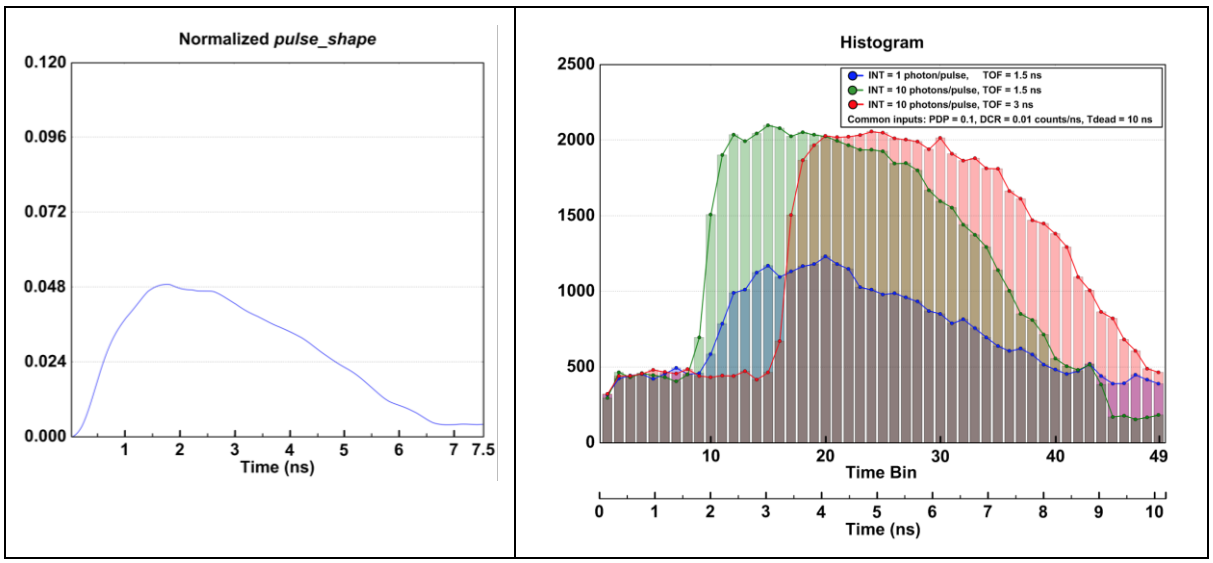


Figure 9. Timing diagram of the SPAD-based sensor when *TOF* = 3 ns and *Tdead* = 10 ns.



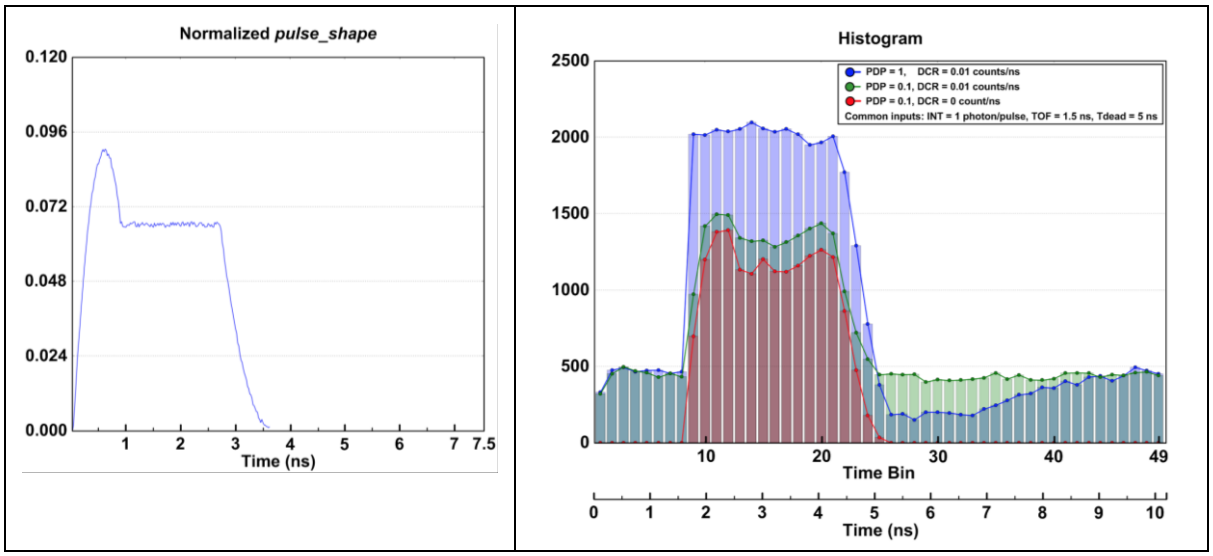


Figure 10. Normalized *pulse_shape* value (left) and measured histogram output with 4,095 measurements (right).
For the first *pulse_shape* (left, top), the remaining inputs were fixed at *PDP* = 0.1, *DCR* = 0.01 counts/ns, and *Tdead* = 10 ns.
For the second *pulse_shape* (left, bottom), the remaining inputs were fixed at *INT* = 1 photons/pulse, *TOF* = 1.5 ns, and *Tdead* = 5 ns.

VI. Conclusion

This work demonstrated the feasibility of system-level simulation of SPAD-based sensors using SystemVerilog. By utilizing the proposed statistical behavioral model of the SPAD, the entire sensor system can be verified, and the histogram result can be predicted in early design and verification stages. We confirmed that the proposed SPAD model and system-level verification platform successfully verified the design of the SPAD-based direct Time-of-Flight sensor.

Acknowledgment

This work was supported by the BK21 FOUR Project.

References

[1] F. Piron, D. Morrison, M. R. Yuce and J. -M. Redouté, "A Review of Single-Photon Avalanche Diode Time-of-Flight Imaging Sensor Arrays," IEEE Sensors Journal, vol. 21, no. 11, pp. 12654-12666, June. 2021.

[2] Becker W. Advanced time-correlated single-photon counting techniques, Springer, 2005.

[3] Scientific Analog, Inc. XMODEL. [Online]. Available at: https://www.scianalog.com/xmodel.

[4] Z. Cheng, X. Zheng, D. Palubiak, M. J. Deen and H. Peng, "A Comprehensive and Accurate Analytical SPAD Model for Circuit Simulation," IEEE Transactions on Electron Devices, vol. 63, no. 5, pp. 1940-1948, May. 2016.

[5] G. Giustolisi, R. Mita and G. Palumbo, "Verilog-A modeling of SPAD statistical phenomena," IEEE International Symposium of Circuits and Systems (ISCAS), Brazil, 2011.

[6] N. A. W. Dutton, S. Gnecchi, L. Parmesan, A. J. Holmes, B. Rae, L. A. Grant and R. K. Henderson, "11.5 A time-correlated single-photon-counting sensor with 14GS/S histogramming time-to-digital converter," IEEE International Solid-State Circuits Conference (ISSCC), 2015.

[7] T. Al Abbas, N. A. W. Dutton, O. Almer, N. Finlayson, F. M. D. Rocca and R. Henderson, "A CMOS SPAD Sensor With a Multi-Event Folded Flash Time-to-Digital Converter for Ultra-Fast Optical Transient Capture," IEEE Sensors Journal, vol. 18, no. 8, pp. 3163-3173, April. 2018.