Multistage Linear Feedback Shift Register Counters With Reduced Decoding Logic in 130-nm CMOS for Large-Scale Array Applications

Daniel Morrison, *Student Member, IEEE*, Dennis Delic, *Member, IEEE*, Mehmet Rasit Yuce, *Senior Member, IEEE*, and Jean-Michel Redouté, *Senior Member, IEEE*

***Abstract*— Linear-feedback shift register (LFSR) counters have been shown to be well suited to applications requiring large arrays of counters and can improve the area and performance compared with conventional binary counters. However, significant logic is required to decode the count order into binary, causing system-on-chip designs to be unfeasible. This paper presents a counter design based on multiple LFSR stages that retains the advantages of a single-stage LFSR but only requires decoding logic that scales logarithmically with the number of stages rather than exponentially with the number of bits as required by other methods. A four-stage four-bit LFSR proof of concept was fabricated in 130-nm CMOS and was characterized in a time-to-digital converter application at 800 MHz.**

***Index Terms*— 3-D imaging, binary counters, decoding logic, event counters, linear-feedback shift register (LFSR), single-photon detection.**

I. Introduction

With recent advances in applications such as single-photon detection, it has become necessary to implement a large number of arrayed counters in small areas. These include time-of-flight (TOF) ranging in depth cameras [1]–[5] where counters are required to count clock cycles and also photon-counting cameras that count the number of photons in an interval [6]–[8]. Reducing the area consumed by the counter in these applications is critical in increasing the number of pixels in the cameras, as each camera pixel contains a separate counter.

While linear-feedback shift registers (LFSRs) are typically used as pseudorandom number generators [9], [10], it has been shown that they are also an efficient way to implement synchronous counters [11] and are well suited to large arrayed designs, as the shift register can act as a serial readout mechanism [12]. LFSR counters have been used in the CMOS pixel design [13] and in single-photon detection arrays [14]. The clock speed of an LFSR is independent of the number of bits in the counter, and they traverse all states in the state space except the all zero state. However, the count order of LFSRs is pseudorandom, so extra processing is required to decode the LFSR state into binary order.

Three different techniques to decode the LFSR sequence into binary are compared in [11]: the iteration method, the direct lookup table (LUT) method, and a time-memory tradeoff algorithm. The iteration method iterates over the entire count sequence of the LFSR and compares each to the counter value. For an n-bit LFSR, this requires approximately 2^{n-1} comparisons on average. The direct LUT method instead uses an $n \times n$ LUT that directly decodes the LFSR state. The time-memory tradeoff algorithm introduced in [11] combines both methods by storing $2^{(N/2)}$ LFSR count values in a table and iterating over the LFSR sequence until the count value matches a value in the table. The number of iterations is then subtracted from the stored value to obtain the decoded value. Another algorithm based on discrete logarithms was introduced in [15] and was adapted for use with ring generator event counters in [16].

Applications with large arrays require every cell in the array to be decoded to binary order for further processing, and for system-on-chip designs, it is necessary to perform this decoding on chip. This requirement dictates that the decoding logic must be integrable and fast, since many conversions need to occur. However, all of the above-mentioned methods grow exponentially in either time or area with the size of the LFSR. For single-photon detection applications, there are several examples of arrayed designs that would not be able to be implemented with LFSR counters without prohibitively large integrated LUTs [17]–[19].

This paper proposes a new counter design based on multiple LFSR stages, which can be decoded with logic that grows logarithmically with the counter size rather than exponentially. While a straightforward concatenation of LFSR counters would cause a significant performance reduction, similar to binary ripple counters, this paper introduces a technique to distribute the ripple signal in time and compensates for this in a generalized decoding logic scheme. This paper also presents

Manuscript received May 24, 2018; revised August 14, 2018; accepted September 17, 2018. Date of publication October 11, 2018; date of current version December 28, 2018. This work was supported by the Australian Government Research Training Program Scholarship. *(Corresponding author: Daniel Morrison.)*

D. Morrison, M. R. Yuce, and J.-M. Redouté are with the Department of Electrical and Computer Systems Engineering, Monash University, Clayton, VIC 3800, Australia (e-mail: daniel.morrison@monash.edu).

D. Delic is with the Defence Science and Technology Group, Edinburgh, SA 5111, Australia.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2018.2872021

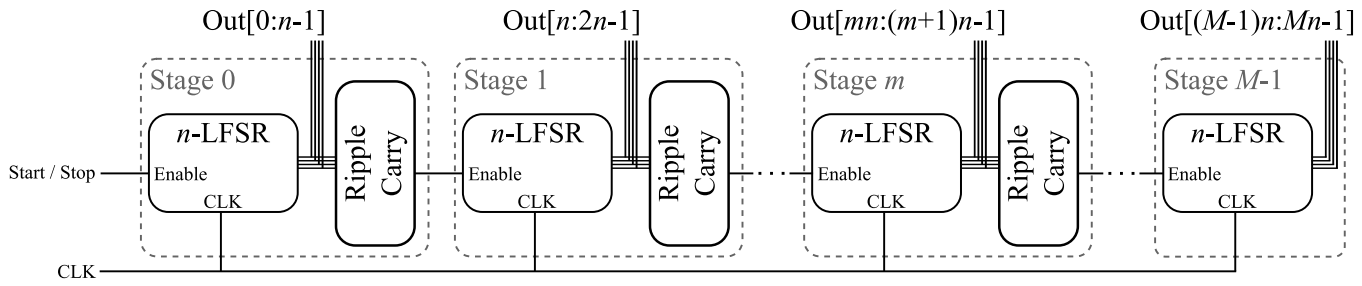

Fig. 1. Block diagram of the multistage LFSR counter.

a proof of concept implementation and characterization of this counter design in a 130-nm CMOS process. Throughout the remainder of this paper, an n-bit LFSR will be referred to as an n-LFSR.

This paper is organized as follows. Section II introduces the proposed counter design, while Section III compares a dynamic logic implementation of the counter to conventional LFSR counters. The fabricated 16-bit counter design is evaluated experimentally and characterized in Section IV. Section V discusses the merits and limitations of the design compared with the state of the art. This paper is concluded in Section VI.

II. Proposed Counter Design

The general scheme of the counter design is shown in Fig. 1. There are M identical n-LFSR blocks that are controlled by an enable signal. When the $(m-1)$th n-LFSR undergoes a specific state change, the enable signal is asserted so that the mth n-LFSR advances one state. This allows the entire $M \times n$ bit state space to be traversed. In large arrayed designs, the counter can also act as a high-speed serial readout chain. This is achieved with minimal additional logic that bypasses the LFSR feedback and ripple-carry blocks.

The multistage counter scheme reduces the counter into M independent modules, allowing each n-LFSR to be decoded separately by an $n \times n$ bit LUT rather than an $(M \times n) \times (M \times n)$ bit LUT. For small n, the LUT can easily be implemented on chip.

A. LFSR Block

Each stage of the counter is triggered once per period of the previous stage, so missing states from the LFSR sequence will cause large blocks of counter states to be missing from the counter state space. Thus, it is important that the n-LFSR is designed for a maximal length. The maximal sequence length of an n-LFSR is only $2^n - 1$, so additional logic is required to incorporate the missing state into the count sequence. This can be achieved using a NOR and XOR function to disable the feedback logic when the `0x000...1` state is detected, as shown in Fig. 2(c). This sequence-extension logic extends the sequence length of the individual component LFSRs to 2^n so that the counter covers every state in the $2^{M \times n}$ state space. This also allows the multistage counter to be used in applications that require every state to be covered, such as self-starting counters, where traditional LFSRs would not be applicable.

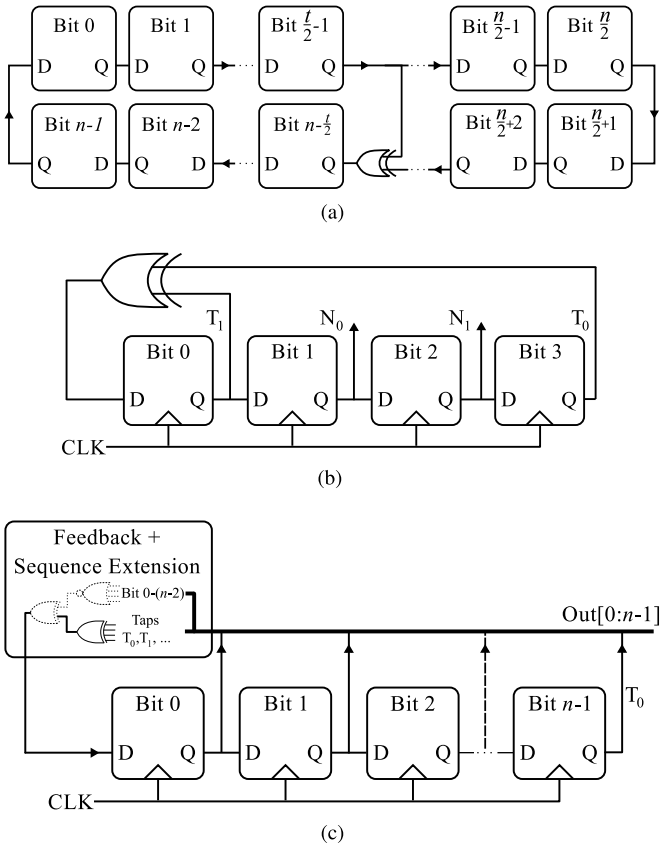

Fig. 2. (a) Structure of a conventional n-bit ring generator. (b) Structure of conventional many-to-one 4-LFSRs. (c) Structure of a proposed multistage n-LFSR block with sequence-extension logic (dotted components). The entire feedback block is implemented as a single logic block.

Several LFSR feedback styles exist, including many-to-one, one-to-many (alternatively known as Fibonacci and Galois LFSRs, respectively), and ring generators. Ring generators [depicted in Fig. 2(a)] are typically regarded as the optimal way to implement an LFSR [20], where the shift register forms a ring and taps form subloops within the ring. However, the sequence-extension requires additional logic in the LFSR, dominating the critical path. Instead, many-to-one style LFSRs [Fig. 2(b)] are used, allowing the feedback logic and the sequence-extension logic to be combined into a single logic block for logic minimization as shown in Fig. 2(c). The multistage counter allows flexibility in choice of the size of the n-LFSR, so that small single-tap LFSRs are preferentially

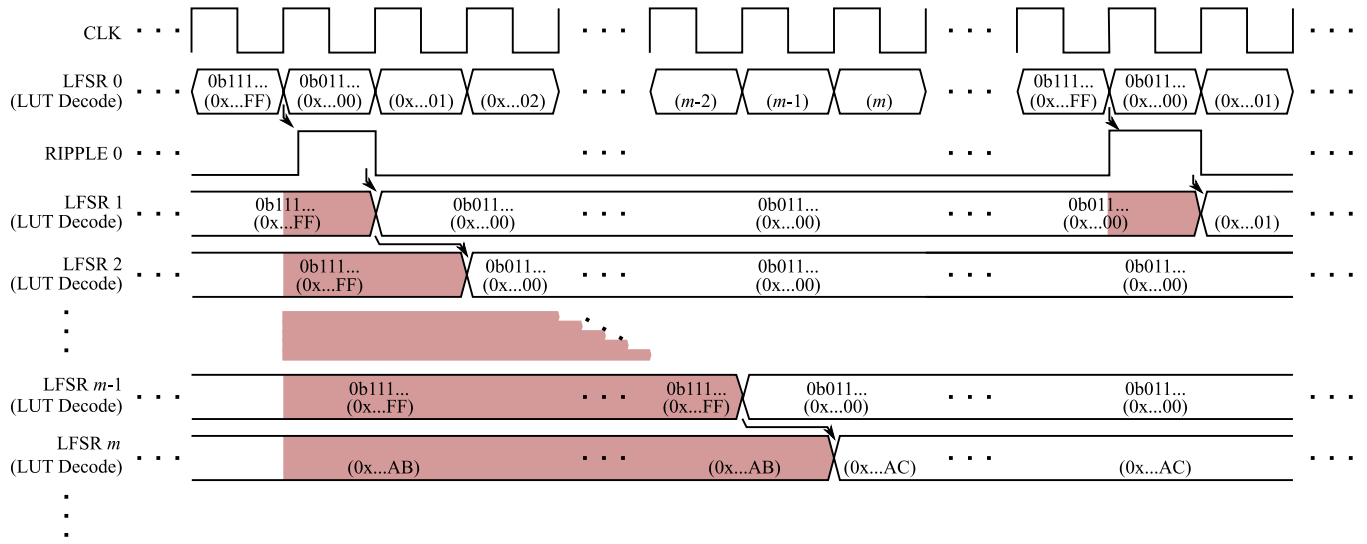

Fig. 3. Timing diagram of the operation of the multistage LFSR counter. Arrows show the operation of the ripple-carry logic. Highlighted states require further processing by the decoding logic.

chosen. A single-tap many-to-one LFSR is topologically indistinguishable from the corresponding ring generator.

B. Ripple-Carry Logic

Since the n-LFSR contains every state in the state space, the LFSR must include the `0b1111...` → `0b0111...` transition. This state is a Gray-code transition and occurs in every n-LFSR design, so it is an ideal ripple trigger transition. This sets the start of the n-LFSR sequence to `0b0111...` so that it is decoded by the decoding logic to `0x...00`.

If the counter was designed so that an LUT could directly decode every stage correctly in a single clock cycle, the ripple signal would need to propagate through every stage and detect if each stage will transition. Instead, to prevent the performance of the counter from decreasing with every extra stage added to the counter, the ripple signal only acts on the direct next stage and the ripple signal for the subsequent stages is carried to the next clock cycle. This distributes the transition edge over time and, for the mth stage, adds an m clock cycle delay to the transition edge.

The counter timing diagram that demonstrates the operation of the ripple-carry logic is shown in Fig. 3. Each LFSR state is given as a binary value (`0b...`), whereas the state after decoding with an LUT (the *LUT Decode* signal) is the hex value in brackets (`0x...`) for each state. When LFSR 0 transitions from the `0b1111...` state to the `0b0111...` state, the RIPPLE 0 signal is generated. On the next clock edge, the RIPPLE 0 signal acts on LFSR 1 causing it to also undergo the `0b1111...` → `0b0111...` transition. This, therefore, also generates a ripple signal to act on LFSR 2 on the next clock edge. In this way, the ripple-carry logic causes the transition edge to be delayed one clock cycle per stage. The delayed transition causes an error triangle to form, shown by highlighted states in Fig. 3. These states are decoded incorrectly by the LUT and therefore need to be corrected with a minor amount of decoding logic in addition to the $n \times n$ bit LUT.

C. Decoding Logic

The decoding logic acts as a postprocessing step on the multistage LFSR counter array output. As each LFSR value is read out of the array, it is passed through an LUT. Fig. 3 shows that the LUT corrects most states to binary order. However, additional logic is required to correct the errors caused by the delayed transition.

Two types of LUT decode errors occur: initial errors and overflow errors. Initial errors occur in the states on the upper edge of the transition error triangle when the counter is stopped on the clock cycle before the mth stage transitions. The decoded value of the previous stages is also the number of clock cycles since the start of the transition edge. Since the ripple takes m clock cycles to reach the mth stage, these errors can be detected if the decoded value of the previous stages is equal to $m-1$. Overflow errors occur when the previous stage has an error and is equal to `0x...FF`. These errors indicate that a previous stage should have caused a ripple event on an earlier clock cycle.

The decoding logic that detects and corrects these errors is shown in Fig. 4. The error detection of each stage depends on the decoded value of the previous stages, so each stage is processed sequentially. If an error condition on the next stage is detected, the *next stage invalid* register is set, so that it is corrected on the next clock cycle. The errors are only ever one less than the correct value, so the *next stage invalid* selects either the LUT output or adds one to the LUT output.

An overflow error in the next stage will occur if the current stage is an error and also `0x...FF`. This can be detected by ANDing the incrementer carryout with the *next stage invalid* register. Initial errors can be detected by storing the previously decoded stages in latches and comparing with a counter that keeps track of the current stage number. If the current stage is equal to the previously decoded value, the next stage will have an initial error. The counter only needs to count to M and therefore needs $y = \lceil \log_2(M) \rceil$ bits. Therefore, only a y-bit comparison needs to be made between the previous decoded

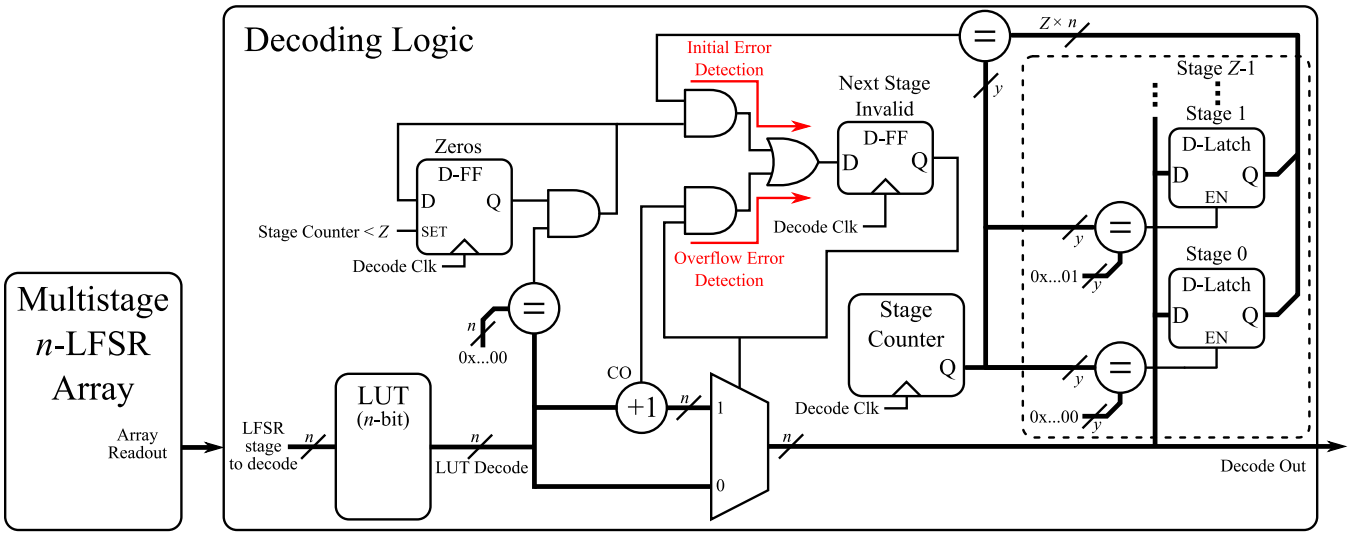

Fig. 4. Logic to decode the multistage LFSR counter state order into binary. Each stage is decoded separately in sequence.

state and the counter, so $Z = \lceil (y/n) \rceil$ stages need to be stored. The *zeros* register is used to ensure that all other bits in the previously decoded value are zero so that the comparison is valid.

The counter zero point (`0x...00`) is an error state. If all counter stages are reset to the zero state, the starting count values will be decoded incorrectly. However, the final state (`0b1111...`) is not an error state, so if the counter is reset to this state instead, the counter will correctly transition to zero after one clock cycle. The final count value will be off by one, but this can be corrected by adding one to the final count value if required.

The time to decode the count value will depend on both M and n. The decoding logic takes one clock cycle per stage of the counter, so M clock cycles are required to decode the entire count value. The critical path of the decoding logic is implementation-dependent but potentially includes the LUT, the incrementer, and the comparison to the stage counter, all of which have reduced performance for large n. Thus, the maximum clock rate will be lower for larger n values, but more stages will be required for an equivalently sized counter with a larger n value, requiring fewer clock cycles to decode the counter. If the readout of the array is performed serially, then the decoding logic can be set up as a pipeline in the readout chain. This only adds an M clock cycle pipeline delay to the entire array readout, but all individual counter values will be read out with a decoded value.

III. Dynamic CMOS Implementation and Comparison

In order to compare the performance and area of the multistage LFSR counter to conventional LFSRs, an implementation was designed to be used as part of a single-shot timing circuit. Some critical factors of the envisioned single-photon detection applications of the design are the number of pixels in the array and the fill factor of the photosensitive area [21]. Both of these factors require that the counter is minimized in the area while maintaining high performance. The ripple-carry logic proposed in Section II-B, if implemented in conventional logic techniques, would require an extra memory element to store the state edge detection in addition to the logic detecting the transition state. This would significantly reduce the area efficiency of the counter. Instead, dynamic logic can be used to combine the state edge detection logic and memory element into the same logic block efficiently (see Section III-B). Dynamic logic also has a tendency to improve the packing density of the transistors [22], so the entire implementation was designed using dynamic logic techniques to minimize the overall counter area.

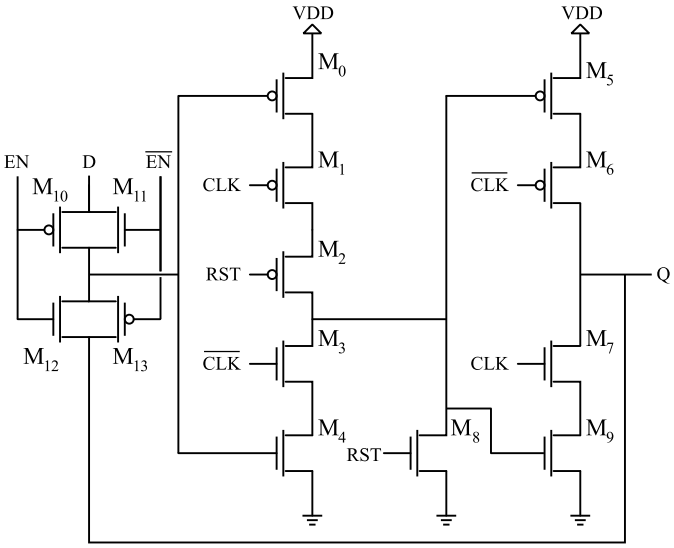

Fig. 5. Schematic of the dynamic flip-flop with reset high.

A. n-LFSR Implementation

The basic flip-flop used in the design is shown in Fig. 5. This is a dynamic flip-flop with extra transistors M_2 and M_8

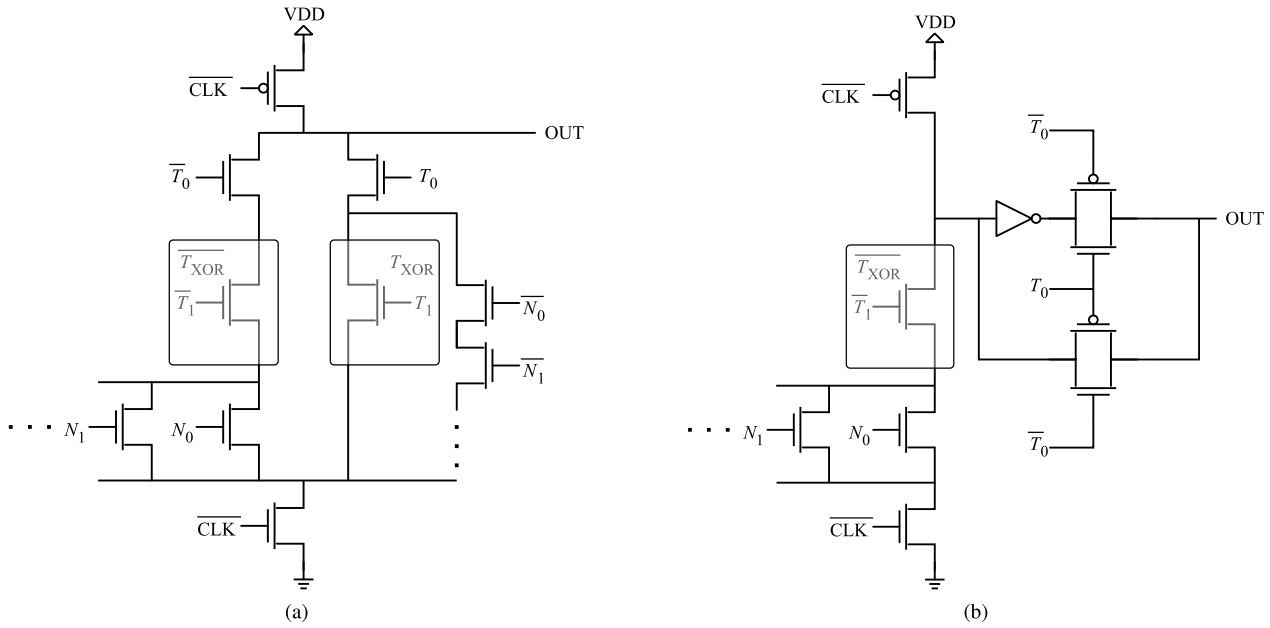

Fig. 6. Schematic of the dynamic feedback logic for maximal length, single-tap, many-to-one n-LFSR with sequence-extension logic. T inputs represent tap bits and N inputs are nontap bits. T_0 is the last bit in the LFSR. (a) $n < 7$. (b) $n \geq 7$. To extend to multiple taps, the T_{XOR} and $\overline{T_{\text{XOR}}}$ blocks should be replaced with the XOR and XNOR of all tap bits, respectively. The branch in (a) evaluating N should also be ANDed with the inverse of all tap bits.

that reset the flip-flop by draining the parasitic capacitor. The flip-flop needs to be constantly refreshed to prevent leakage from the parasitic capacitor discharging the state. M_{10}–M_{13} were added as transmission gates from the output to the input to allow the flip-flop to be clocked without changing state. The n-bit shift register is formed by chaining n flip-flops. The output of the flip-flop is only driven, while the clock is high, so a buffer is required for each flip-flop to ensure that the D input is always driven when the enable signal changes.

Using the notation illustrated in Fig. 2(b) and (c), T_0 as the final LFSR bit, T_x as the xth tap of the LFSR, T_{XOR} as the XOR of all LFSR taps (except T_0), and $\bar{N}$ as the NOR of all nontap bits, a dynamic feedback logic block required for a many-to-one LFSR with 2^n states can be implemented using the following nMOS pull-down network:

$$\bar{D} = T_0(T_{\text{XOR}} + \bar{T}_1\bar{T}_2 \ldots \bar{N}) + \bar{T}_0(\{\overline{T_{\text{XOR}}} - \bar{T}_1\bar{T}_2 \ldots\} + \bar{T}_1\bar{T}_2 \ldots N) \quad (1)$$

as presented in the Appendix. Fig. 6(a) shows the transistor implementation of this feedback network for a single-tap LFSR. As n becomes large, the series connection of all N bits becomes slow and the number of transistors required to implement the feedback becomes large. For large n, it becomes more efficient to directly evaluate

$$D = T_0 \oplus (T_{\text{XOR}} + \bar{T}_1\bar{T}_2 \ldots \bar{N}) \quad (2)$$

using a dynamic logic block to evaluate $T_{\text{XOR}} + \bar{T}_1\bar{T}_2 \ldots \bar{N}$ and a transmission gate-based XOR gate. This implementation, shown in Fig. 6(b), only depends on the parallel combination of all nontap bits (N), removing the series connection of all nontap bits ($\bar{N}$) that was required to implement (1). However, the extra stage implementing the XOR gate adds a delay to the evaluation of the feedback logic. It was shown in simulation that Fig. 6(b) becomes the more efficient implementation in both area and performance for $n \geq 7$.

The complexity of these two logic blocks scale with the number of taps in the n-LFSR, so the tap configuration that gives a maximal sequence length with the least number of taps is preferred to minimize the number of transistors in the feedback logic. The LFSRs used in the following comparisons use the tap configurations provided in [23].

B. Ripple-Carry Logic Implementation

The ripple-carry logic detects when the n-LFSR undergoes the `0b1111...` → `0b0111...` transition and signals the next stage to increment one state on the next clock edge. A general implementation of the ripple-carry logic is shown in Fig. 7(a). The circuit operation is shown in Fig. 7(b).

1) The ripple clock is generated from the main clock so that the rising edge occurs before the rising edge of the main clock by some delay (the ripple delay).
2) While the ripple clock is high and all bits in the LFSR are high, then the center node will be pulled low. This detects that the state before the clock edge was `0b1111...`
3) On the rising edge of the main clock, the LFSR advances to the next state and BIT_0 will be driven low. This causes the output node to be pulled high and the `0b1111...` → `0b0111...` transition has been detected.
4) On the falling edge of the clock, the center node is reset.
5) On the next rising clock edge, the output is reset low.
6) The ripple signal is the enable for the next LFSR stage allowing it to increment one state after the state transition.

Transistor M_5 prevents the next stage from triggering, while the counter is stopped. This implementation is targeted as a

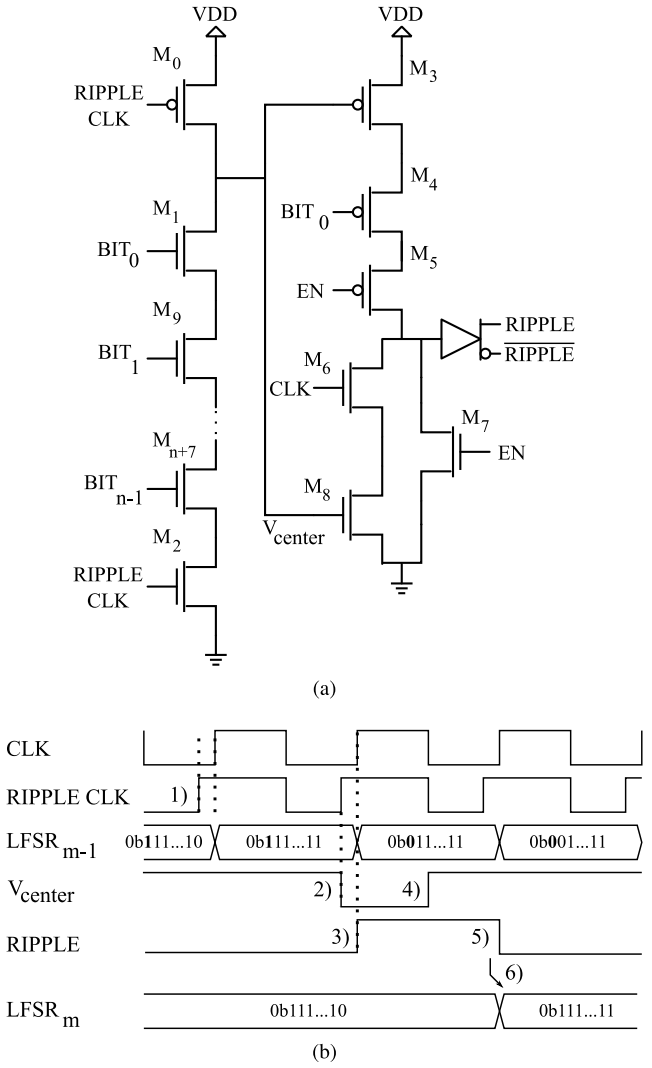

Fig. 7. (a) Schematic of the logic that detects the `0x...FF` to `0x...F7` state edge of the n-LFSR to ripple the next stage. (b) Timing diagram of the operation of the ripple-carry logic.

single-shot counter, so it needs to be reset after being stopped to prevent missing states from the count order. This design can be extended to a complete implementation by adding extra logic to store the detection of the `0x...FF` state, while the counter is stopped.

C. Compared Architectures

Throughout the remainder of this section, comparisons will be made between the multistage LFSR counter and the various conventional architectures listed in the following. While these architectures could be implemented using standard cells or conventional logic techniques, for a fair comparison, each architecture is implemented using the same dynamic logic style and flip-flops as the multistage LFSR architecture.

1) 2-LFSR: The 2-LFSR is a special case of an LFSR where a maximal length feedback function can be implemented using a single inverter between the output of one flip-flop and the input of the other. Similarly, a 1-LFSR can be implemented in the same way.

2) Ring Generator: The ring generator is implemented, as shown in Fig. 2(a). The flip-flops are arranged in a ring, and each tap forms a subloop within the ring. The subloop tap is XORed with the main ring so that the number of flip-flops to the left of the subloop is the same as the tap number for a maximal length LFSR [20]. The ring generator represents the ideal performance of an n-LFSR in the comparison.

3) Many-to-One LFSR With Sequence-Extension Logic: The many-to-one LFSR is implemented in the same way as a single block of the multistage LFSR architecture, as described in Section III-A. When compared with the ring generator, this architecture reveals the performance and area impact of the sequence-extension logic extending the LFSR count sequence to 2^n. A further comparison with the multistage LFSR shows the effect of the ripple-carry logic.

4) Binary Counter: The binary counter comparison is implemented using a hierarchical Manchester carry chain as described in [11].

D. Area Comparison

The number of transistors used by each type of LFSR counter, using the same logic design techniques, are calculated throughout this section using the notation, A_x as the number of transistors used by block x, n as the number of bits in the LFSR, and t as the number of taps (excluding T_0) required for a maximal sequence in the LFSR.

1) 2-LFSR: The 2-LFSR uses only an inverter and two flip-flops, so the size of this LFSR is therefore

$$A_{2\text{LFSR}} = 2A_{\text{FF}} + 2. \tag{3}$$

2) Ring Generator: The ring generator requires a flip-flop for each bit and a 2-bit XOR gate per tap. Hence

$$A_{\text{ring}} = nA_{\text{FF}} + tA_{2-\text{XOR}}. \tag{4}$$

3) Many-to-One LFSR With Sequence-Extension Logic: The many-to-one LFSR requires a flip-flop for each bit, and the feedback logic is implemented as in Fig. 6. The number of transistors required in the feedback logic is the size of implementing the XOR function as well as the transistors required for the sequence-extension logic. A multiple-bit wide XOR pull-down network can be implemented in a recursive tree structure requiring

$$A_{t-\text{XOR}} = 2 + 4 + \cdots + 2^t + 2^t \tag{5}$$

$$= 2^t + 2\sum_{k=0}^{t-1} 2^k \tag{6}$$

$$= 3 \cdot 2^t - 2 \text{ transistors}. \tag{7}$$

Overall, the feedback logic implemented in a single stage (for $2 < n < 7$) requires

$$A_{\text{fb1}} = 3 \cdot (2^t - 1) + 2n - t \text{ transistors} \tag{8}$$

and the two-stage feedback logic implementation requires

$$A_{\text{fb2}} = n - t - 1 + A_{2-\text{XOR}} + 3 \cdot 2^{(t-1)} \text{ transistors}. \tag{9}$$

The overall size of the many-to-one LFSR is

$$A_{\text{mto}} = nA_{\text{FF}} + \begin{cases} 3\cdot(2^t-1)+2n-t, & 2<n<7 \\ n-t-1+A_{2-\text{XOR}}+3\cdot 2^{(t-1)}, & n\geq 7. \end{cases} \tag{10}$$

4) Multistage LFSR Counter: The multistage LFSR counter design uses M, many-to-one n-LFSR stages with the ripple-carry logic for each. Fig. 7(a) shows that the ripple-carry logic is implemented using

$$A_{\text{ripple}} = n + 12 \text{ transistors.} \tag{11}$$

Therefore, the total size of the multistage LFSR counter is

$$A_{\text{multistage LFSR}} = (m-1)(n+12) + m\cdot A_{\text{mto}}(n). \tag{12}$$

5) Binary Counter: The hierarchical Manchester carry chain [11] uses a maximum of 4 bits per carry chain and an AND gate to detect the carry skip conditions for the higher level carry chains. Each carry chain element uses three transistors with buffers for each carry output. An XOR gate is also required for each bit of the counter.

A comparison of the number of transistors used by each counter implementation relative to the number of transistors required by the flip-flops is shown in Fig. 8(a). This provides a measure of additional area required by logic other than the flip-flops, where a relative area of 1 implies that the counter has no additional logic.

E. Performance Comparison

Similar to the comparison made in Section III-D, this section compares the performance of the multistage LFSR counter to the many-to-one LFSR with sequence-extension logic, the ring generator, and the binary counter. The multistage LFSR implementation has five potential critical paths.

1) The voltage on the gates of the feedback logic must settle when the clock is high.
2) The precharge of the feedback logic must complete when the clock is high.
3) The feedback logic output to the flip-flop input must settle when the clock is low.
4) The ripple-carry logic center node must be pulled low within the ripple delay.
5) The ripple signal propagation to the next stage must settle within a clock period.

Among these critical paths, 3)–5) depend on the size of the LFSR, but all are independent of the number of stages. Increasing the size of the LFSR will impact the performance but adding extra stages will not, allowing the size of the counter to be scaled without reducing the performance.

The many-to-one LFSR critical path is the same as the multistage LFSR without the ripple-carry logic and thus is defined by critical paths 1)–3). The ring generator critical path only consists of the XOR gate between two flip-flops and the fan-out of a flip-flop to one or two gates depending on the number of adjacent taps. The critical path of the hierarchical Manchester carry chain occurs during the `0xFFF...` → `0x000...` transition where the carry is required to propagate through all levels of the carry chain.

All of these critical paths were simulated to compare the performance over counter size. Fig. 8(b) shows the simulation results of the maximum clock frequency of each counter type, normalized to the performance of the 2-LFSR. This gives a measure of the performance reduction from the maximum performance that an LFSR can achieve to each compared architecture.

The multistage LFSR counter was limited by the feedback logic for the cases of $n < 9$ other than the special case of $n = 2$. In these cases, the performance of the multistage LFSR counter is similar to the performance of the corresponding many-to-one n-LFSR but is able to keep the performance constant over much larger counter sizes by adding stages. The cases of $n \geq 9$ are impacted by the series connection of all LFSR bits in the ripple-carry logic. Since the multistage LFSR counter requires small n-LFSR stages, the impact of the ripple-carry logic will not affect a practical implementation.

The critical path of the many-to-one LFSR (with sequence extension) was limited by the sequence-extension logic rather than the structure of the LFSR. Thus, comparing the ring generator to many-to-one LFSR in Fig. 8(b) shows the increased delay caused by the extension of the sequence length to 2^n. The performance of the sequence-extension logic is also impacted by the number of taps used in the many-to-one style feedback. This is especially prevalent in the 8-LFSR, where the increased complexity of the three-tap feedback logic causes the performance to be worse than the single-tap 9-LFSR. These slowdowns are mitigated in the multistage LFSR architecture by preferentially using small, single-tap n-LFSR blocks.

F. Decoding Logic Comparison

The size of the multistage LFSR counter decoding logic can be compared with the decoding methods of conventional LFSRs in [11] by estimating the number of transistors required by each method. The iteration method is not feasible with large arrays, so this method will be ignored. The direct LUT method uses an $n \times n$ LUT. A single-bit wide pass-transistor LUT [24] requires two transistors for every branch. The total number of transistors required is given by

$$T_{\text{LUT}_1\times n} = 2+4+8+\cdots+2^n \tag{13}$$

$$= 2\sum_{k=0}^{n-1} 2^k = 2^{n+1}-2. \tag{14}$$

An extra copy of the LUT is required for each bit of the counter, so the full LUT required for the direct method requires

$$T_{\text{LUT}_{\text{direct}}} = n(2^{n+1}-2). \tag{15}$$

The time-memory tradeoff algorithm in [11] needs to be converted to a CMOS implementation to be compared with the other methods. This can be achieved using a comparison element for each of the $2^{(n/2)}$ entries in the stored table, followed by pass transistors to multiplex the correct table number into a subtracter. A $\lceil (n/2) \rceil$ bit counter is also needed

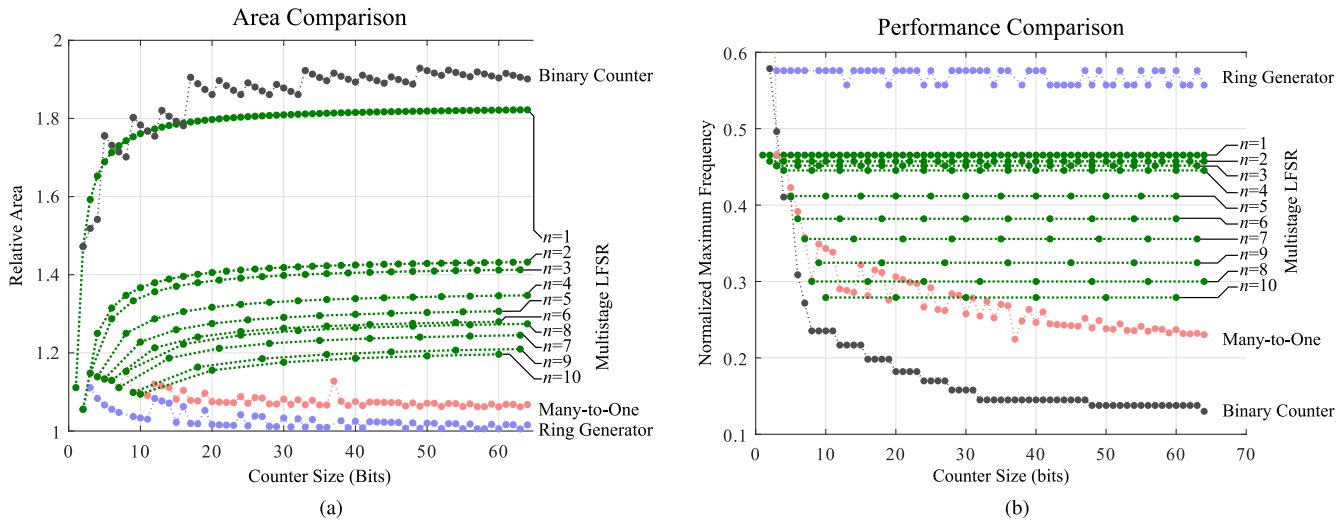

Fig. 8. (a) Comparison of area used by the different LFSR counter schemes. Relative area is the ratio of transistors used in the counter to the number of transistors used by the flip-flops. (b) Simulation results of the performance of LFSR counters. The performance is normalized to the maximum clock rate of the 2-LFSR. Thus, 2-LFSRs have a normalized performance of 1 but are not shown on the plot.

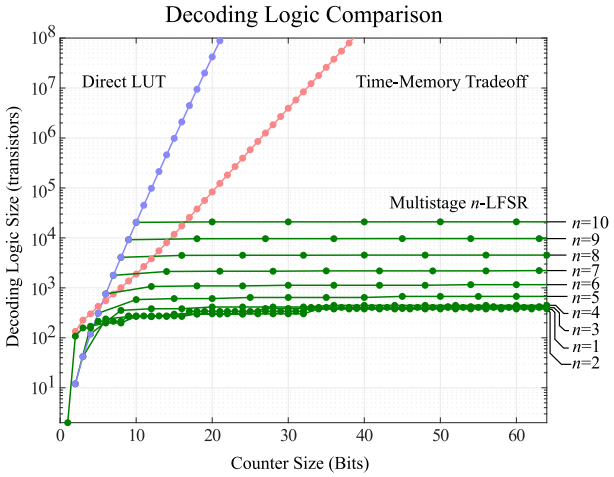

Fig. 9. Comparison between the decoding logic required by the multistage LFSR, the direct LUT method, and the time-memory tradeoff method [11].

to count the iterations, and an n-bit LFSR is required to generate the count sequence.

The multistage LFSR decoding logic (Fig. 4) is constant for a given n, other than the latches that store the previously decoded stages and the size of the stage counter. The stage counter needs y bits ($\lceil \log_2(M) \rceil$), and Z latches ($\lceil (y/n) \rceil$) are required to decode the M stages. Thus, the size of the decoding logic for a multistage LFSR will scale proportionally to $\log_2(M)$. The remaining decoding logic size was calculated using the number of transistors required to implement the logic blocks using conventional CMOS logic techniques for $2 \leq n \leq 10$.

While an $n \times n$ LUT is required as part of the decoding logic of the multistage LFSR counter, if n is kept small and extra stages are used to extend the counter size, the decoding logic does not scale exponentially as occurs in the other methods. Fig. 9 shows the comparison between the different methods.

TABLE I

COMPARISON OF DECODING LOGIC PROCESSING TIME

Method	Steps	Frequency Limitation
Direct LUT	1	Total number of bits (LUT Size)
Time-Memory Tradeoff [11]	$2^{\left(\frac{N}{2}-1\right)} \cdot \left(\frac{N}{2}\right)$ [(a)]	Total number of bits (Counter Size)
Multistage n-LFSR	M	n (LUT Size)

[(a)] Average decoding time.

There is always a multistage LFSR configuration that has a smaller decoding logic compared with the other methods.

A comparison table of the decoding time for each method is shown in Table I. The direct LUT method only requires a single processing step. However, the minimum time required to perform this step will depend on the number of branches in the LUT which scales exponentially with the number of bits in the LFSR. Conversely, the multistage n-LFSR requires M processing steps to decode the counter, but each step will be considerably shorter than an equivalent $(n \times M)$-LFSR counter decoded by the direct LUT method. Each step of the multistage n-LFSR decoding method depends on an $n \times n$ LUT. The minimum time required for each processing step in the time-memory tradeoff method presented in [11] is primarily limited by the speed of the counter and is therefore much faster than both the multistage n-LFSR decoding method and the direct LUT method. However, the number of processing steps scales exponentially with the size of the LFSR and the processing time is nondeterministic.

When specifically decoding a large-scale array of counter values, the decoding logic can be introduced as part of a serial readout scheme. For the multistage n-LFSR, if the decoding logic is part of the readout pipeline, only M clock cycles will be added to the readout of the entire array.

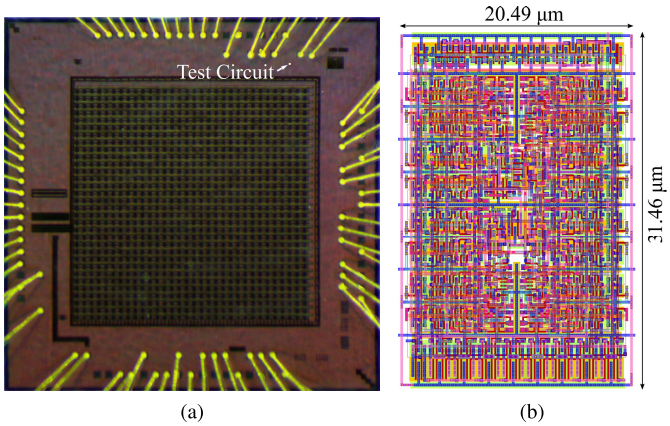

Fig. 10. (a) Die microphotograph. (b) Layout of the four-stage 4-LFSR.

IV. Four-Stage 4-LFSR Counter Measurements

A four-stage 4-LFSR counter was incorporated in the design of a 16-bit time-to-digital converter (TDC) as an example application of the multistage LFSR counters. The TDC was fabricated in 130-nm HV CMOS and used in a 32 × 32 pixel TOF camera system as shown in the die microphotograph in Fig. 10(a).

The four-stage 4-LFSR was based on the dynamic logic implementation presented in Section III using the feedback logic depicted in Fig. 6(a). In addition to the four-stage 4-LFSR counter, the TDC also includes an SR-latch and synchronization stage to enable and disable the counter based on start and stop signals and a clock buffer to drive the clock transistors and generate the ripple clock. The clock buffer is a nonoverlapping tapered clock generator, and the ripple clock waveform was generated using a nonsymmetric tapered buffer. While the counter design is intended to be read out serially, for testing purposes, the TDC output is read out of the array in parallel using tristate inverting buffers. Overall, the TDC is implemented in an area of 20.49 μm × 31.46 μm and the layout is shown in Fig. 10(b).

A. Postlayout Simulations

Postlayout simulations of the implemented four-stage 4-LFSR were performed to determine the actual critical path and the effect of PVT variations on the performance of the design. The performance was determined by measuring the slack in each potential critical path identified in Section III-E and then calculating the minimum clock required to satisfy each path. While the performance analysis showed that the critical path is limited by the feedback logic for the $n = 4$ case, additional parasitic capacitance caused the postlayout critical path to be the propagation of the ripple-carry signal to the next stage. The implemented design was optimized for the area rather than performance, so additional buffers to overcome the parasitic capacitance could be used to improve the performance of the design.

Plots of the performance over process corners are shown in Fig. 11(a) for supply voltage variations and in Fig. 11(b) for temperature variations. These plots show that the design

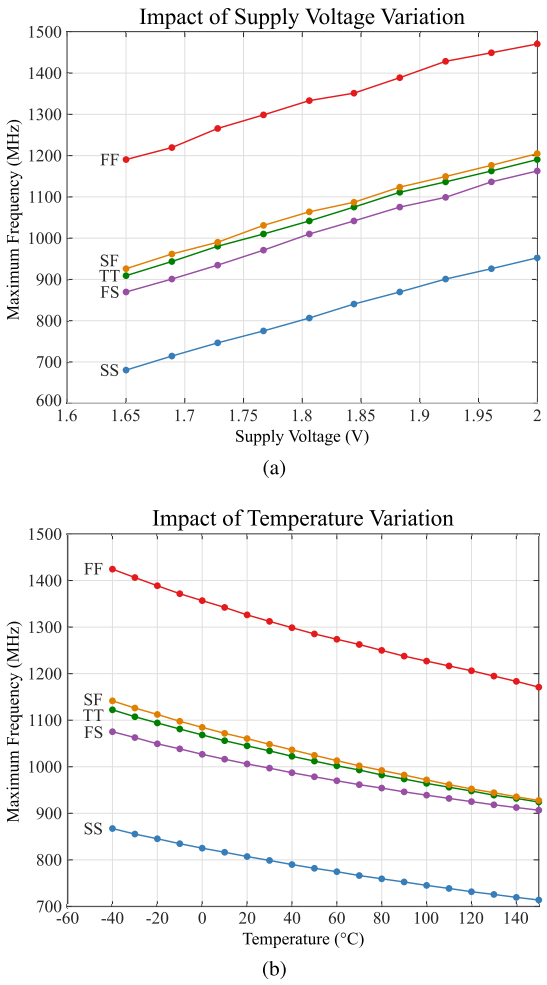

Fig. 11. (a) Impact of supply voltage variation on the maximum operating frequency of the four-stage 4-LFSR over process corners. (b) Impact of temperature variation on the maximum operating frequency of the four-stage 4-LFSR over process corners. Process corner labels are specified nMOS first.

is more heavily impacted by slow pMOS corners than slow nMOS corners. This is due to the critical path involving the evaluation through M_3–M_5 in the ripple-carry logic [Fig. 7(a)].

Table II shows the breakdown of the power consumption of each block from the postlayout simulations at 600 MHz, with the counter consuming an average power of 157.5 μW under the continuous operation. This shows that the first stage of the counter uses significantly more power than the later stages. Thus, the power consumption could be reduced by optimizing the first stage of the counter for low power consumption without impacting the high density of the dynamic logic design of later stages. This would not be possible in a regular LFSR design where every bit is active on every clock cycle.

B. Counter and Decoding Logic Experimental Validation

To test the TDC and counter design, the chip was connected to an Opal Kelly ZEM4310 to generate the control signals and to read out test data to a computer.

The inverted output of each stage of the counter running at 500 kHz is shown in the oscilloscope capture in Fig. 12. This shows that the 4-LFSRs traverse all 16 states in the

TABLE II
POSTLAYOUT POWER CONSUMPTION SIMULATIONS AT 600 MHz

Block	**Device Power Consumption (μW)**		
Counter	LFSR	Ripple	Subtotal
Stage 0	113.4	8.046	121.5
Stage 1	24.79	1.095	25.88
Stage 2	8.547	0.085	8.632
Stage 3	1.450	N/A	1.450
Counter Total			157.5
External			
Clock Buffer	543.7		
Line Driver	0.031		
Input Conditioning	0.341		
Parasitic	46.40		
Total Power Consumption	747.9		

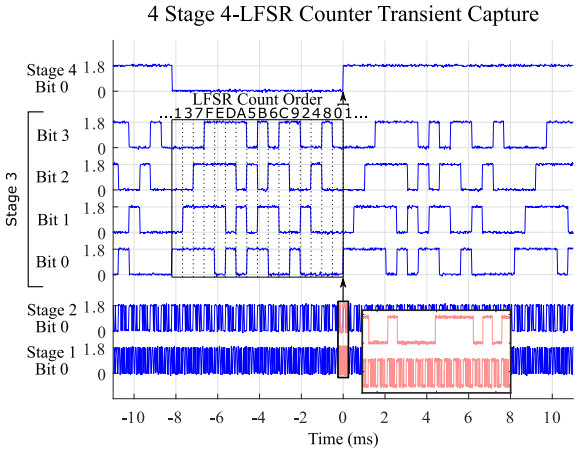

Fig. 12. Oscilloscope capture of the four-stage 4-LFSR counter operation at 500 kHz.

expected order and that the ripple-carry logic correctly causes subsequent stages to progress exactly one state after a `0x0` to `0x1` state transition.

To verify the performance at a higher frequency, the time between the start and stop signals was incremented in approximately 200-ps steps and compared with the counter output to get a measure of the timing error of the TDC. The output of the counter was decoded using a field-programmable gate array implementation of the decoding logic in Fig. 4. Both the raw output of the counter and the decoded output were transmitted to a computer where the raw output was decoded directly using an LUT to verify the operation of the decoding logic.

The plot in Fig. 13 shows that the maximum timing error of the TDC is 0.84 LSB at 800 MHz; 58 outlier data points of the 413 686 measured data points were identified with timing errors between 1.66 and −5.58 LSB. All of these outliers occurred on a state transition and were attributed to an issue with the synchronization stage of the TDC causing a metastable state on the hold signal of the counter. This was confirmed in simulation and measurement.

The TDC was verified to operate up to a maximum clock frequency of 880 MHz. This slowdown from the simulated

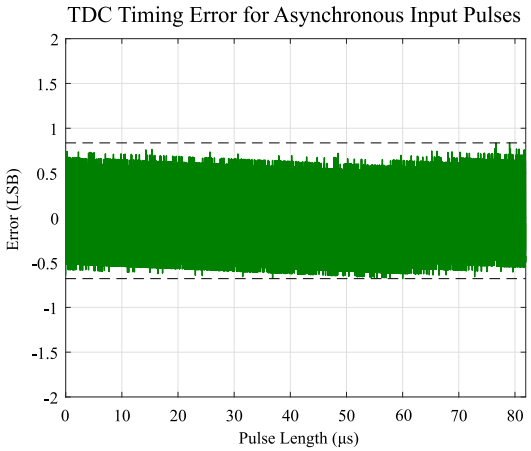

Fig. 13. Timing error of the four-stage 4-LFSR TDC at 800 MHz over the entire state space of the counter. The input pulses were incremented in approximately 200-ps steps and were asynchronous to the counter clock.

results in Section IV-A is due to the design of the clock buffer driving the counter. Dynamic logic is used in the design, so there is a minimum operating frequency caused by capacitor leakage. The TDC was verified to operate with a frequency of 10 kHz, which is significantly lower than the operating frequencies in the reported designs for the intended applications (see Section V).

C. Power Consumption Measurement

The power consumption of the implemented TDC was measured using a 10-s average current measurement. Fig. 14 shows the power consumption over clock frequency. The power measurements match the postlayout simulations well with the total power consumption measured at 600 MHz being 799 μW and simulated as 747.9 μW. It should be observed that most of the power is consumed by circuitry external to the counter (Table II). This power consumption can be reduced using a low-power clock buffer design [25] or by sharing the clock buffer among multiple cells.

V. DISCUSSION

A comparison of the area, power consumption, and operating frequency of the implemented four-stage 4-LFSR to other state-of-the-art single-photon detection arrays is presented in Table III. This shows that the four-stage 4-LFSR compares well in terms of area to other single-photon sensor arrays, although in comparing area, it is important to note that the compared designs may include other circuitry within the stated area. The four-stage 4-LFSR also allows a much higher clock frequency than the reported counter operating frequencies of the other designs allowing the counter of the TDC to have a higher timing resolution.

The continuous power consumption of the four-stage 4-LFSR is high compared with the power consumption of the other designs partially due to the usage of dynamic logic. However, the TDCs in these applications are typically operated at a low duty cycle [2], [4]. When this is considered,

TABLE III
COMPARISON TO PRIOR WORK

Ref:	This Work	[2]	[3]	[4]	[5]	[14]
Technology	130 nm CMOS	180 nm CMOS	150 nm CMOS	130 nm CMOS	180 nm CMOS	350 nm CMOS
Type	Counter Only	Image Sensor	Image Sensor	Linear Array	Image Sensor	Image Sensor
Application	Arrayed Designs	TOF [a] / PC [b]	TOF [a] / PC [b]	TOF [a]	TOF [a]	TOF [a] / PC [b]
Average Power	799 µW[c] / 157.5 µW[d] (Counter)	9 µW (TDC)	47.7 mW (Entire Array)	15 µW (TDC)	530 mW (Entire Array)	50 mW (Entire Array)
Power Measurement Duty Cycle (%)	Continuous	0.5	Unspecified	<10	Unspecified	Unspecified
Area (µm)	20.49 × 31.46 (Counter)	29 × 28 (TDC)	60 × 60 (Pixel)	64 × 47 (Pixel)	Unspecified	150 × 150 (Pixel)
Counter Operating Frequency (MHz)	880	Unspecified	100	246	100	N/A
Bits	16	11	16	12	12	18
Array Size	N/A	64 × 64	64 × 64	1 × 400	202 × 96	64 × 32

[a] Time-of-Flight
[b] Photon Counting
[c] Measured at 600 MHz. Includes clock buffer power consumption.
[d] Simulated at 600 MHz. Counter in isolation.

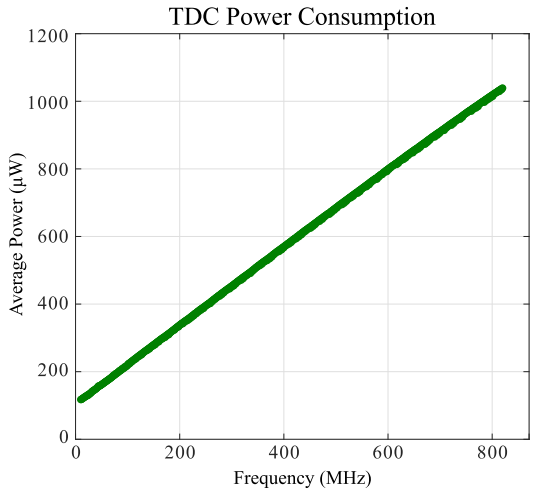

Fig. 14. Power consumption measurement of the four-stage four-LFSR TDC.

the average power consumption of the four-stage 4-LFSR is comparable with the other designs.

This paper was implemented in a 130-nm CMOS process as it is typical in single-photon detection applications. Extending this design to more advanced technology nodes for other applications would require care due to the dynamic logic style used. Dynamic logic is inherently more susceptible to noise than conventional logic techniques [26], and the increased PVT variation may cause further performance variability [27].

Compared with the conventional LFSR counter designs, the multistage n-LFSR has a performance and area penalty due to the additional sequence-extension and ripple-carry logic required. However, if the counter value is required to be decoded on chip, the multistage n-LFSR provides a method to convert the LFSR count order into binary in an efficient way. Overall, the multistage n-LFSR architecture provides a tradeoff between the high performance and area efficiency of the LFSR counters with the decoding process required to convert the sequence into binary.

VI. CONCLUSION

This paper presents a generalized design and a practical implementation in 130-nm CMOS of multistage LFSR counters as well as the decoding logic required to convert the count sequence into binary order. The proposed counter is composed of multiple smaller LFSR stages that are triggered by a specific state transition of the previous stage. This configuration allows the decoding logic to be based on a constant sized LUT for any number of stages, rather than requiring the LUT to scale with the size of the counter. The decoding logic of the proposed counter scales proportionally to the logarithm of the number of stages, rather than exponentially with the number of bits as required by decoding methods for conventional LFSRs. The multistage LFSR counter retains many of the same advantages that LFSR counters possess, such as high performance independent of the number of bits in the counter at the tradeoff of a small amount of extra logic.

A proof of concept of this counter design used in an integrated TOF camera application was fabricated in 130-nm CMOS and was verified to operate as expected with a maximum timing error of 0.84 LSB at 800 MHz. The multistage LFSR can provide both the performance and area benefits of LFSR counters to any application that requires an array of event counters, such as single-photon imaging sensors, while also allowing for decoding logic that can be implemented on chip.

An extension of this paper would be to generalize this multistage counter design to allow different types of counters in different stages using the same ripple-carry technique. A generalized multistage counter could use the high performance of the LFSR counter for the first stage while using binary counters for subsequent stage potentially allowing performance, area, or power consumption opportunities in the future counter designs.

APPENDIX

The feedback function required to obtain a maximal length LFSR is an XOR function between the final bit output and several taps within the LFSR [28]. Furthermore, to ensure that the all zero state is included as part of the sequence, the NOR of all bits of the LFSR except the final bit is also XORed into the feedback function

$$D = T_0 \oplus T_{\text{XOR}} \oplus (\bar{T_1}\bar{T_2}\ldots\bar{N}). \tag{16}$$

Both T_{XOR} and $(\bar{T_1}\bar{T_2}\ldots\bar{N})$ depend on the tap bit states, allowing (16) to be minimized. Expanding the second XOR in (16) gives

$$T_{\text{XOR}} \oplus (\bar{T_1}\bar{T_2}\ldots\bar{N}) = T_{\text{XOR}}\overline{(\bar{T_1}\bar{T_2}\ldots\bar{N})} + \overline{T_{\text{XOR}}}(\bar{T_1}\bar{T_2}\ldots\bar{N}). \tag{17}$$

The first term in (17) can be reduced by noting that

$$T_{\text{XOR}}\overline{(\bar{T_1}\bar{T_2}\ldots\bar{N})} = T_{\text{XOR}}(T_1 + T_2 + \cdots + N) \tag{18}$$

and since $(T_1 + T_2 + \cdots)$ is only false when all T_x are false, a case not present in T_{XOR}

$$T_{\text{XOR}}(T_1 + T_2 + \cdots + N) = T_{\text{XOR}} + T_{\text{XOR}}N = T_{\text{XOR}}. \tag{19}$$

The second term in (17) can also be minimized as the $\bar{T_1}\bar{T_2}\ldots$ annihilates all terms in $\overline{T_{\text{XOR}}}$ except the case where all taps are false. Therefore,

$$\overline{T_{\text{XOR}}}(\bar{T_1}\bar{T_2}\ldots\bar{N}) = \bar{T_1}\bar{T_2}\ldots\bar{N}. \tag{20}$$

The feedback function can be implemented as a single dynamic logic block with a pull-down evaluation stage by implementing the inverse of (16) in nMOS logic. Using the simplifications in (19) and (20) and expanding the XNOR in the inverse of (16) give

$$\bar{D} = T_0(T_{\text{XOR}} + \bar{T_1}\bar{T_2}\ldots\bar{N}) + \bar{T_0}\overline{(T_{\text{XOR}} + \bar{T_1}\bar{T_2}\ldots\bar{N})}. \tag{21}$$

$\overline{(T_{\text{XOR}} + \bar{T_1}\bar{T_2}\ldots\bar{N})}$ can be simplified similar to (21)

$$\begin{aligned}
&\overline{(T_{\text{XOR}} + \bar{T_1}\bar{T_2}\ldots\bar{N})} \\
&\quad = \overline{T_{\text{XOR}}}(T_1 + T_2 + \ldots + N) && (22)\\
&\quad = \overline{T_{\text{XOR}}}(T_1 + T_2 + \cdots) + \overline{T_{\text{XOR}}}N && (23)\\
&\quad = \{\overline{T_{\text{XOR}}} - \bar{T_1}\bar{T_2}\ldots\} + \overline{T_{\text{XOR}}}N && (24)\\
&\quad = \{\overline{T_{\text{XOR}}} - \bar{T_1}\bar{T_2}\ldots\}(1 + N) + \bar{T_1}\bar{T_2}\ldots N && (25)\\
&\quad = \{\overline{T_{\text{XOR}}} - \bar{T_1}\bar{T_2}\ldots\} + \bar{T_1}\bar{T_2}\ldots N. && (26)
\end{aligned}$$

Using (26) in (21) gives

$$\begin{aligned}
\bar{D} = {} & T_0(T_{\text{XOR}} + \bar{T_1}\bar{T_2}\ldots\bar{N}) \\
& + \bar{T_0}(\{\overline{T_{\text{XOR}}} - \bar{T_1}\bar{T_2}\ldots\} + \bar{T_1}\bar{T_2}\ldots N).
\end{aligned} \tag{27}$$

REFERENCES

[1] D. Bronzi, Y. Zou, F. Villa, S. Tisa, A. Tosi, and F. Zappa, "Automotive three-dimensional vision through a single-photon counting SPAD camera," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 3, pp. 782–795, Mar. 2016.

[2] I. Vornicu, R. Carmona-Galán, and A. Rodríguez-Vázquez, "A CMOS 0.18 μm 64×64 single photon image sensor with in-pixel 11 b time-to-digital converter," in *Proc. Int. Semiconductor Conf. (CAS)*, 2014, pp. 131–134.

[3] M. Perenzoni, D. Perenzoni, and D. Stoppa, "A 64×64-pixels digital silicon photomultiplier direct TOF sensor with 100-MPhotons/s/pixel background rejection and imaging/altimeter mode with 0.14% precision up to 6 km for spacecraft navigation and landing," *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 151–160, Jan. 2017.

[4] J. M. Pavia, M. Scandini, S. Lindner, M. Wolf, and E. Charbon, "A 1×400 backside-illuminated SPAD sensor with 49.7 ps resolution, 30 pJ/sample TDCs fabricated in 3D CMOS technology for near-infrared optical tomography," *IEEE J. Solid-State Circuits*, vol. 50, no. 10, pp. 2406–2418, Oct. 2015.

[5] C. Niclass, M. Soga, H. Matsubara, M. Ogawa, and M. Kagami, "A 0.18-μm CMOS SoC for a 100-m-range 10-frame/s 200×96-pixel time-of-flight depth sensor," *IEEE J. Solid-State Circuits*, vol. 49, no. 1, pp. 315–330, Jan. 2014.

[6] R. Ballabriga, M. Campbell, E. Heijne, X. Llopart, L. Tlustos, and W. Wong, "Medipix3: A 64 k pixel detector readout chip working in single photon counting mode with improved spectrometric performance," *Nucl. Instrum. Methods Phys. Res. A, Accel. Spectrom. Detect. Assoc. Equip.*, vol. 633, pp. S15–S18, May 2011.

[7] J. Kim, S. Park, M. Hegazy, and S. Lee, "Comparison of a photon-counting-detector and a CMOS flat-panel-detector for a micro-CT," in *Proc. IEEE Nucl. Sci. Symp. Med. Imag. Conf. (NSS/MIC)*, 2013, pp. 1–4.

[8] F. Villa *et al.*, "SPAD smart pixel for time-of-flight and time-correlated single-photon counting measurements," *IEEE Photon. J.*, vol. 4, no. 3, pp. 795–804, Jun. 2012.

[9] H. Mo and M. P. Kennedy, "Masked dithering of MASH digital delta-sigma modulators with constant inputs using multiple linear feedback shift registers," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 64, no. 6, pp. 1390–1399, Jun. 2017.

[10] K. J. Sham, S. Bommalingaiahnapallya, M. R. Ahmadi, and R. Harjani, "A 3×5-Gb/s multilane low-power 0.18-μm CMOS pseudorandom bit sequence generator," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 55, no. 5, pp. 432–436, May 2008.

[11] A. Ajane, P. M. Furth, E. E. Johnson, and R. L. Subramanyam, "Comparison of binary and LFSR counters and efficient LFSR decoding algorithm," in *Proc. IEEE 54th Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2011, pp. 1–4.

[12] S. Soh *et al.*, "16 bit multi-energy level detecting photon counting ROIC," in *Proc. IEEE Nucl. Sci. Symp. Med. Imag. Conf. Rec. (NSS/MIC)*, Oct./Nov. 2012, pp. 801–804.

[13] M. Kłosowski, W. Jendernalik, J. Jakusz, G. Blakiewicz, and S. Szczepański, "A CMOS pixel with embedded ADC, digital CDS and gain correction capability for massively parallel imaging array," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 64, no. 1, pp. 38–49, Jan. 2017.

[14] D. Bronzi *et al.*, "100 000 frames/s 64×32 single-photon detector array for 2-D imaging and 3-D ranging," *IEEE J. Sel. Topics Quantum Electron.*, vol. 20, no. 6, pp. 354–363, Nov. 2014.

[15] D. W. Clark and L.-J. Weng, "Maximal and near-maximal shift register sequences: Efficient event counters and easy discrete logarithms," *IEEE Trans. Comput.*, vol. 43, no. 5, pp. 560–568, May 1994.

[16] N. Mukherjee, A. Pogiel, J. Rajski, and J. Tyszer, "High-speed on-chip event counters for embedded systems," in *Proc. 22nd Int. Conf. VLSI Design*, 2009, pp. 275–280.

[17] T. A. 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 J.*, vol. 18, no. 8, pp. 3163–3173, Apr. 2018.

[18] M.-A. Tetrault, E. D. Lamy, A. Boisvert, R. Fontaine, and J.-F. Pratte, "Low dead time digital SPAD readout architecture for realtime small animal PET," in *Proc. IEEE Nucl. Sci. Symp. Med. Imag. Conf. (NSS/MIC)*, Oct. 2013, pp. 1–6.

[19] N. Krstajić *et al.*, "A 256 × 8 SPAD line sensor for time resolved fluorescence and raman sensing," in *Proc. 40th Eur. Solid-State Circuits Conf. (ESSCIRC)*, 2014, pp. 143–146.

[20] N. Mukherjee, J. Rajski, G. Mrugalski, A. Pogiel, and J. Tyszer, "Ring generator: An ultimate linear feedback shift register," *IEEE Comput.*, vol. 44, no. 6, pp. 64–71, Jun. 2011.
[21] D. Bronzi, F. Villa, S. Tisa, A. Tosi, and F. Zappa, "SPAD figures of merit for photon-counting, photon-timing, and imaging applications: A review," *IEEE Sensors J.*, vol. 16, no. 1, pp. 3–12, Jan. 2016.
[22] V. Friedman and S. Liu, "Dynamic logic CMOS circuits," *IEEE J. Solid-State Circuits*, vol. JSSC-19, no. 2, pp. 263–266, Apr. 1984.
[23] P. Alfke, "Efficient shift registers, LFSR counters, and long pseudo-random sequence generators," Xilinx Inc., San Jose, CA, USA, Tech. Rep. XAPP 052, Jul. 1996.
[24] R. Cunha, H. Boudinov, and L. Carro, "Quaternary look-up tables using voltage-mode CMOS logic design," in *Proc. 37th Int. Symp. Multiple-Valued Logic (ISMVL)*, 2007, p. 56.
[25] C. Yoo, "A CMOS buffer without short-circuit power consumption," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 47, no. 9, pp. 935–937, Sep. 2000.
[26] L. Ding and P. Mazumder, "On circuit techniques to improve noise immunity of CMOS dynamic logic," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 12, no. 9, pp. 910–925, Sep. 2004.
[27] M. Alioto, G. Palumbo, and M. Pennisi, "Understanding the effect of process variations on the delay of static and domino logic," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 5, pp. 697–710, May 2010.
[28] K. Martin, "Digital integrated system building blocks," in *Digital Integrated Circuit Design*. New York, NY, USA: Oxford Univ. Press, 2000, pp. 407–408.

Daniel Morrison (S'16) received the B.E. degree (honors) in electrical and computer system engineering and the B.Sc. degree in physics and applied mathematics from Monash University, Clayton, VIC, Australia, in 2015, where he is currently working toward the Ph.D. degree with the BICS SPAD Group.

He is currently a Researcher with the BICS SPAD Group, Monash University. His current research interests include VLSI digital systems, single-photon detection, and integrated sensor front ends.

Dennis Delic (M'07) received the B.E. degree (honors) in electronic engineering and the Ph.D. degree from the University of South Australia, Adelaide, SA, Australia, in 1993 and 2002, respectively.

From 1996 to 2000, he was a Senior Bipolar Design Engineer with Philips Semiconductors, where he was responsible for the design of data interface and power control application-specific integrated circuits. From 2001 to 2006, he was with Integrated Device Technologies, where he was designing CMOS advanced programmable switching and timing products. In 2006, he joined the Defence Science and Technology Group, where he is involved in the scientific research of advanced technologies and manages a broad portfolio of technical projects. He currently leads the development of high-density CMOS single-photon avalanche diode (SPAD) arrays. His current research interests include SPAD-based flash LADAR for both bathymetric and 3-D imaging applications.

Dr. Delic received the DST Fellowship to research SPAD image sensors in 2012.

Mehmet Rasit Yuce (S'01–M'05–SM'10) received the M.S. degree in electrical and computer engineering from the University of Florida, Gainesville, FL, USA, in 2001, and the Ph.D. degree in electrical and computer engineering from North Carolina State University, Raleigh, NC, USA, in 2004.

He was a Postdoctoral Researcher with the Electrical Engineering Department, University of California at Santa Cruz, Santa Cruz, CA, USA, in 2005. He was an Academic Member with the School of Electrical Engineering and Computer Science, University of Newcastle, Callaghan, NSW, Australia, until 2011. In 2011, he joined Monash University, Clayton, VIC, Australia, where he is currently an Associate Professor with the Department of Electrical and Computer Systems Engineering. He has authored the books *Wireless Body Area Networks* in 2011 and *Ultra-Wideband and 60 GHz Communications for Biomedical Applications* in 2013. His current research interests include wearable devices, Internet-of-Things for health care, wireless implantable telemetry, wireless body area network, biosensors, and integrated circuit technology dealing with digital, analog, and radio-frequency circuit designs for wireless, biomedical, and RF applications. He has published more than 150 technical articles in these areas.

Dr. Yuce received the NASA Group Achievement Award in 2007 for developing a silicon-on-insulator transceiver, the Best Journal Paper Award from the IEEE Microwave Theory and Techniques Society in 2014, and the Research Excellence Award from the Faculty of Engineering and Built Environment, University of Newcastle, in 2010. He is a Topical Editor of the IEEE SENSORS JOURNAL and was a Guest Editor of the IEEE JOURNAL OF BIOMEDICAL AND HEALTH INFORMATICS in 2015.

Jean-Michel Redouté (M'09–SM'12) received the M.S. degree in electronics from the University of Antwerp, Antwerp, Belgium, in 1998, the M.Eng. degree in electrical engineering from the University of Brussels, Brussels, Belgium, in 2001, and the Ph.D. degree from the University of Leuven, Leuven, Belgium, in 2009. His Ph.D. thesis was on the design of EMI resisting analog integrated circuits.

In 2001, he was with Alcatel Bell, Antwerp, where he was involved in the design of analog microelectronic circuits for telecommunications systems. In 2005, he joined the ESAT-MICAS Laboratories, University of Leuven, as a Ph.D. Research Assistant. In 2009, he was a Post-Doctoral Scholar with the Berkeley Wireless Research Center, University of California at Berkeley, Berkeley, CA, USA. In 2010, he joined Monash University, Clayton, VIC, Australia, as a Senior Lecturer. His current research interests include mixed-signal integrated circuit (IC) design, electromagnetic compatibility, biomedical (IC and non-IC) circuit design, and radio-frequency IC design.