Multi Stage Counters using high performance Linear-feedback Shift registers (LFSR)

Belamala Naidu, Sudhansu Sekhar Behera

1M.Tech Student Department of ECE, MVGR College of Engineering, Vizianagaram, AP, India.
2Asst.Prof Department of ECE, MVGR College of Engineering, Vizianagaram, AP, India.

Abstract—Applications like single-photon detection, it is necessary to implement large arrayed counters which takes small area. In such applications, Linear-feedback shift register (LFSR) counters have been shown to be well suited to applications requiring large arrays of counters. These counters can improve the area and performance compared with conventional binary counters. However, significant logic is required to decode the count order of LFSR into binary order which takes additional logic circuitry. This paper presents a Multistage counter design which uses LFSR counters as well as binary counters. Combination of these two counters gives a better improvement in power and takes same area like existing counter. Decoding logic used in this counter scales logarithmically with the number of stages rather than exponentially with the number of bits as required by other methods. 4 stage multistage counter is implemented in this paper on Xilinx Vivado 2014.2 version.

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

INTRODUCTION

According to recent advances in applications such as single-photon detection, it has become necessary to implement a large number of arrayed counters in small areas. Single photon detection means essentially the detection and timing of a single photoelectron released by light from a photosensitive surface of a photomultiplier, channel multiplier, avalanche multiplier photodiode, or other photo devices. These include time-of-flight (TOF) ranging in depth cameras [1] where counters are required to count clock cycles and also photon-counting cameras that count the number of photons in an interval [2]. The Time-of-Flight principle (ToF) is a method for measuring the distance between a sensor and an object, based on the time difference between the emission of a signal and its return to the sensor, after being reflected by an object. 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.

Linear-feedback shift registers (LFSRs) are used as pseudorandom number generators [3], [4], it has been shown that they are also an efficient way to implement synchronous counters [5] and are well suited to large arrayed designs, as the shift register can act as a serial readout mechanism [6]. LFSR counters have been used in the CMOS pixel design and in single-photon detection arrays [7]. 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 [5]: 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. The direct LUT method instead uses an n*n LUT that directly decodes the LFSR state. This algorithm uses the decimal equivalent of the LFSR state as an index to an array. The direct LUT method, in which a particular LFSR state is generated, is stored in the array indexed by that LFSR state. The time-memory tradeoff algorithm introduced in [5] 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.

This paper proposes a new counter design based on multiple stages in which first stage is LFSR counter and for succeeding stages binary counters are used. These can be decoded with logic that grows logarithmically with the counter size rather than exponentially. While a straightforward concatenation of stages of counters would cause a significant performance reduction. This paper introduces a ripple carry technique to distribute the ripple signal in time and compensates for this in a generalized decoding logic scheme. This ripple carry is placed between two counters to increase performance of multi stage counter. 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 existing multi-stage counter design, while section III introduces the proposed counter design. Comparison between existing counter and proposed counter is given in Section IV. This paper is concluded in Section V.

SECTION-II
EXISTING COUNTER DESIGN

The general scheme of the counter design is shown in Fig. 1. There are 4 identical n-LFSR blocks that are controlled by an enable signal. When the n-LFSR in one stage undergoes a specific state change, the enable signal is asserted so that the n-LFSR in next stage advances one state. This allows the entire 4*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*n bit LUT rather than an $(M*n)*(M*n)$ 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. 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.

one-to-many (alternatively known as Fibonacci and Galois LFSRs, respectively), and ring generators. Ring generators are typically regarded as the optimal way to implement an LFSR [9], 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 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. The multistage counter allows flexibility in choice of the size of the n-LFSR, so that small single-tap LFSRs are preferentially chosen.

B. Ripple-Carry Logic

Since the n-LFSR contains every state in the state space, the LFSR must include the 1111 1110 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 0111 so that it is decoded by the decoding logic to 0000.

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.

![Fig. 3. Timing diagram of the operation of the multistage LFSR counter](image)

The counter timing diagram that demonstrates the operation of the ripple-carry logic is shown in Fig. 3. When LFSR 0 transitions from the 1111 state to the 1110 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 1111 1110 transition. This 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. 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 × n bit LUT.
C. Decoding Logic

The decoding logic acts as a post-processing step on the multistage LFSR counter array output. As each LFSR value is read out of the array, it is passed through an LUT. 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 = [log2(M)] bits. Therefore, only a y-bit comparison needs to be made between the previously decoded state and the counter, so Z = (y/n) 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 (0000) 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 (1111) 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.
Fig. 4. Logic to decode multistage LFSR counter into binary order. Each stage is decoded separately in sequence.

Multi stage counter with decoding logic waveform is shown in fig. 5. The outputs coming from the each n-LFSR is fed to the decoding logic individually. Counter functioning is depend upon the ripple carry logic output. If ripple carry output is 1 then only the next stage LFSR counter is functioning. The role of ripple carry logic is understood by observing the fig. 5.

Fig. 5. multistage LFSR counter with decoding logic waveform

SECTION-III

PROPOSED COUNTER DESIGN

Proposed counter could use the high-performance of the LFSR counter for the first stage while using binary counters for the subsequent stages. Because of this arrangement change in existing multi stage counter leads to potentially allowing performance, area, or power consumption. 4 stage multistage counter design is proposed in this paper which
constitute of one LFSR and three binary counters. In the first stage LFSR counter is placed and in subsequent stages random counters are used. In between two counters ripple carry logic is placed to reduce the performance degradation. Ripple-carry logic uses the output values of respected counter. The proposed counter is shown in fig.6.

![Proposed multistage counter design](image)

The outputs coming from counter in multistage counter are pseudo random. In single photon applications it is necessary to convert LFSR pseudo random count into binary order. Decoding logic done this work. Decoding logic which is used in existing multi stage counter is used in this proposed counter also. Decoding logic used in this paper takes less comparisons to convert LFSR count values into binary order compared with former decoding techniques.

A. n-LFSR IMPLEMENTATION

The basic flip-flop used in the designing a n-LFSR is a D flip-flop. D means data whatever the data present at the side of input that is coming out to output side. The D flip-flop is widely used. It is also known as a "data" or "delay" flip-flop. The D flip-flop captures the value of the D-input at a definite portion of the clock cycle (such as the rising edge of the clock). That captured value becomes the Q output. At other times, the output Q does not change. The D flip-flop can be viewed as a memory cell, a zero-order hold, or a delay line.

Linear feedback shift registers have been the most popular devices to date for generating and handling pseudorandom sequences. LFSR’s usually used as a event counters and pseudorandom number generators. In applications of LFSR, feedback plays major role in generating binary sequences. If XOR gate is used as a feedback element then all zero state is illegal whereas XNOR is used as a feedback then all state is illegal means that state is not present in binary sequence. The maxium sequence length of LFSR is $2^{(n-1)}$ states. In order to
get full sequence length additional gates are required. To get full sequence length, one XOR
gate and one 3 input NOR gate are required.
The sequence length of a LFSR is also depend on tap bits. Those bits are influence input of a
LFSR counter, those bits are called as a tap bits. Bits which are not effecting the input of
counter consider as a non-tap bits. These tap bits are depend upon the feedback elements in
which order they connected in between flip-flops or shift register.

Therefore, sequence length of a LFSR counter also varies in which order feedback
elements are connected. 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 [10].
n-LFSR counter diagram is given in fig. 2.

B. RIPPLE-CARRY LOGIC IMPLEMENTATION

The ripple-carry logic detects when the n-LFSR undergoes the 1111 to 1110 transition
and signals next stage to increment one state on the next clock edge. While a straightforward
concatenation of stages of counters would cause a significant performance reduction. This
paper introduces a ripple carry technique to distribute the ripple signal in time and
compensates for this in a generalized decoding logic scheme. This ripple carry is placed
between two counters to increase performance of multi stagecounter. ANDing the all bits
coming from counter is simple ripple carry logic implementation. When all bits coming from
counter are 1’s then only next stage counter runs otherwise it is not functioning.
The ripple carry operation is given below
i). while the ripple carry is high and all bits in the LFSR are high, this detects that the state
before the clock edge was 1111.
ii). On rising edge of the main clock, the LFSR advances to the next state and next stage
counter is runs.
iii). The ripple signal is the enable for the next LFSR stage allowing it to increment one state
after the state transition.

C. RANDOM COUNTER

Random counter is a one type of binary counter. Apart from remaining binary counter
it has special function. All binary counter like decade counter, up counter synchronous
counter etc., are generates numbers in a binary order where as random counter generates
numbers randomly means not in binary order. This type of counters are useful in testing. In
testing, depend upon number of inputs the possible output combinations are increased or
decreased. So, in these conditions testing all possible outcomes takes more time to test a
particular component or a device. Whenever a pseudo random number generator is used it
saves testing time of a designer.

The basic component to implement a counter is d flipflop. For 4-bit counter 4 D-flip-
flops are taken. output of one flipflop is given as input to the next flipflop. One XOR gate
used in feedback. Outputs of 1&2 flip-flops are taken as inputs of XOR gate. XORed output
is given to the input of 1st flip-flop.
PROPOSED COUNTER WITH DECODING LOGIC

In this paper, multi stage counter is designed with one LFSR counter and three random counters. In between two counters, ripple carry logic is placed to reduce the degradation of data. Ripple carry logic also plays a vital role in functioning of counter. LFSR counter and random counter both are gives a random counter output. These random outputs are again given to the decoding logic which is used to convert the random numbers into binary order. In decoding logic, LUT convert the most of bits into binary order. When the number of bits of multi stage counter, then decoding done by LUT doesn’t gives correct output. So some additional logic is used to convert the bits in binary order which is shown in fig. 4.

Proposed multi stage counter is designed and simulated in Xilinx Vivado 14.2 version. The output waveforms are shown in fig. 7. In the fig .8, initially LFSR counter is functioning. When the LFSR counter output is 1111, then the ripple carry logic is high. When the ripple carry logic is high the the random counter starts counting. Like that following counters are starts counting.

Fig .7. Proposed multi stage counter with decoding logic waveform

SECTION – IV
COMPARISON

The proposed multi stage counter and existing multi stage counter both are functionally verified by using Xilinx Vivado14.2 version. In terms of area both counters are utilizing same area. Coming to the power reports of both counters, proposed counter consumes less power compared to existing multi stage counter. Existing counter consumes power of 4.184watts whereas proposed counter consumes 3.488 watts of power. Decoding logic comparisons is shown in fig .8. In this diagram, proposed decoding logic is takes less area than previous
decoding logic which are given in introduction.

Fig. 8. Comparison between the decoding logic required by the multistage LFSR, the direct LUT method, and the time-memory tradeoff method [5].

SECTION V
CONCLUSION

This paper presents a generalized design and a practical implementation of multistage counter as well as the decoding logic required to convert the count sequence into binary order. The proposed counter is composed of LFSR stages and binary counters 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 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. Random counters were takes less area compared to LFSR counters so in the following stages in multistage counter, random counters are used. Grouping LFSR counter with random counters in proposed counter design takes same area like existing counter and takes less power consumption compared to existing counter design.

An extension of this paper would be to generalize this multistage counter design to increase length of counters in each stages using the same ripple-carry technique.

REFERENCES


Author’s profile:

Belamala Naidu has received his B.Tech Degree in Electronics & Communication Engineering from Avanthi’s St.Teresa college of Engineering and Technology affiliated to JNTU Kakinada in 2017 and pursuing M.Tech degree in (VLSI) MVGR College of engineering(A), Vizianagaram, AP, India.

Sudhansu Sekhar Behera, has received his M.Tech degree in Microelectronics and VLSI from NIT durgapur. He is dedicated to teaching field from the last 2 years. He has guided 2 P.G and 15 U.G students. At present he is working as Assistant Professor in MVGR college of engineering (A), Vizianagaram, AP, India.