Compact FPGA Implementation of Popcounter for BNN Using Linear Feedback Shift Register

Nagisa ISHIURA Ryota SAIMYOJI

School of Science and Technology, Kwansei Gakuin University
2-1 Gakuen, Sanda, Hyogo, 669–1337, Japan

Abstract— This paper presents an idea for reducing the size of neurons for inference of binarized neural networks with a view to pack as many neurons as possible in FPGAs to increase parallelism. We focus on input-serial neurons and use LFSRs (linear feedback shift registers) in place of binary counters. We also presents an idea to use LFSRs in parallel-serial hybrid neurons. Implemented on Xilinx Artix-7, the number of LUTs is reduced by 68.8% and 12.1% for a serial neuron and a hybrid neuron of parallelism 16 , respectively, for threshold 3840.

I. Introduction

Following the recent success of neural network technologies, deep learning is now widening the scope of its applications. One of the trends in this area is the shift from cloud to edge. Especially, neural inference on edge devices is expected to further expand application domain of machine learning.

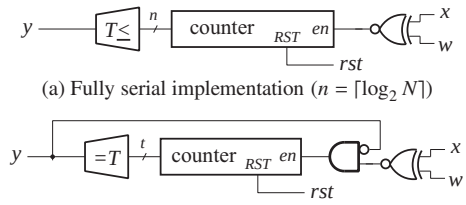
With this in view, active researches are being conducted on developing efficient hardware schemes for neural network inference. A binarized neural network (BNN), a neural network with binary weights and activations [1], is one of the promising schemes to reduce hardware size and many efficient hardware architectures for BNNs have been proposed so far [2, 3, 4].

Since practical neural networks consist of a huge number of neurons which in principle run in parallel, there is abundant parallelism to exploit in hardware design. Then, how to implement neuron hardware efficient both in speed and in area is a key issue. In the case of FPGA implementation, to increase "operations per second per LUT" is one of the design goals.

In this paper, we present an idea for reducing the size of neurons in BNNs. We focus on input-serial neurons and replace an ordinary binary popcounters with LFSRs. We also presents an idea of using LFSRs in the parallel-serial hybrid neurons. Implementation on a Xilinx Artix-7 FPGA shows that the number of LUTs is reduced by 68.8% and 12.1% for a serial neuron and 16-parallel hybrid neuron, respectively, for threshold 3840.

II. Binarized Neural Network (BNN)

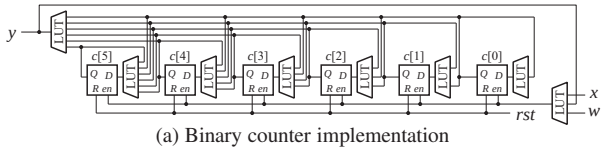
Let N be the number of inputs to a neuron in a BNN. Let x_i and w_i in $\{-1, +1\}$ be the activation value and the weight for the i-th input, respectively, $b \in \mathcal{Z}$ be the bias of the neuron. The activation value y of the neuron is expressed as $y = f(b + \sum_{i=1}^{N} w_i \cdot x_i)$, where $f(x) = 1$ if $0 \le x$ and $f(x) = -1$ otherwise. In the inference phase, we can represent -1 and $+1$ by 0 and 1, respectively, then multiplication for x_i and w_i is reduced to an exclusive-NOR operation. Hereafter, for notational simplicity, we denote the binary representation of x_i and w_i by simply x_i and w_i. The computation of f is translated into counting of the number of 1's in $\{\overline{x_i \oplus w_i} \mid 1 \le i \le N\}$ and examining if it is greater than or equal to threshold T, where $T = (N + b)/2$.



(a) Fully serial implementation ($n = \lceil \log_2 N \rceil$)

(b) Economized implementation ($t = \lceil \log_2 T \rceil$)

Fig. 1. Input-serial Neuron.



(a) Binary counter implementation

(b) LFSR implementation

Fig. 2. LUT-based Implementation of serial Neuron.

In an input-serial neuron, $\overline{x_i \oplus w_i}$ is computed one by one for each i and the number of 1's are counted in N steps, while in a parallel-serial hybrid neuron, P inputs are processed in parallel to complete the computation in N/P steps. The serial neurons are the slowest, but their small implementation allows exploiting other dimensions of parallelism, such as the neuron parallelism or the feature map parallelism.

Fig. 1 (a) shows an example design of a serial neuron. We assume that x_i and w_i are supplied serially from registers or memories. The counter is n-bit wide where $n = \lceil \log_2 N \rceil$ and its value is compared against T. We also assume that T is *embedded* [4], that is, each neuron works for a fixed threshold. Fig. 1 (b) shows a slightly economized design. It stops the increment as soon as the count reaches T, which reduces the necessary bits of the counter to t, where $t = \lceil \log_2 T \rceil$. When T is greater than $N/2$, we may count 0's instead of 1's, so we can reduce the counter size at least by 1 bit.

III. LFSR Implementation of Popcounters

A. Serial Neuron

Fig. 2 (a) shows LUT-based implementation of a serial neuron using a t-bit binary counter ($t = 6$ in this case). Since the

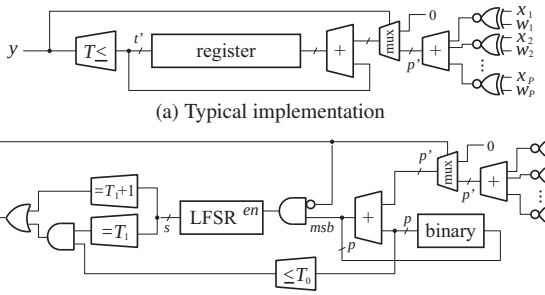


Fig. 3. Parallel-serial hybrid popcounter.

next value of c[i] (the i-th bit of the counter) depends on c[0] through c[i], at least t LUTs are necessary. More LUTs will be necessary for larger t, for c[i]'s function has $i+1$ inputs which may exceed the maximum input number of an LUT.

We propose to replace the binary counter by a Fibonacci LFSR. This results in a design in Fig. 2 (b) where the primitive polynomial is $x^6 + x + 1$. c[1] through c[5] need no LUT and c[0] only one LUT for exclusive-ORing c[5] and c[0]. Since n-th order primitive polynomials with 3 or 5 terms exist for large n, a single LUT is enough to compute the feedback to c[0].

B. Parallel-Serial Hybrid Neuron

Fig. 3 (a) illustrates a structure of a hybrid neuron where P inputs are processed at a time. P exclusive-NORs are computed and resulting 1's are counted by an adder. The adder is p'-bit wide where $p' = \lceil \log_2(P+1) \rceil$. The counts are accumulated into a t'-bit wide register, where $t' = \lceil \log_2(T+P-1) \rceil$, and accumulation is disabled when the count reaches the threshold.

Since LFSRs does not fit directly in the parallel-serial implementation, we propose an LFSR-binary hybrid implementation. The proposed design is illustrated in Fig. 3 (b). The register is divided into the lower p bits and the upper s bits, where $p = \lceil \log_2 P \rceil$ and $s = t' - p$. While the lower part is a binary counter, the upper part is an LFSR. T_0 and T_1 are the lower p bits and the upper s bits of T. The design in (b) is almost equivalent to that in (a) when the LFSR is replaced by an ordinary counter. Thus reduction of the LUT count on the upper part is expected by the use of the LFSR.

IV. Preliminary Result

We implemented neuron modules in Verilog HDL based on the proposed scheme, which are synthesized targeting Xilinx Artix-7 using Xilinx Vivado (2016.4).

Table I (a) summarizes the result for the serial neurons. Note that the circuit size depends only on T but not on N. #cycle shows the number of execution cycles. In the case of serial neurons, it takes $N+1$ cycles (one cycle for initialization). The number of the flip-flops is $\lceil \log_2 T \rceil$ for the all schemes. Our design consumed fewer LUTs, as was suggested in Fig. 2. A neuron for $T = 3840$ can be implemented with only 5 LUTs, which is 68.8% fewer than the binary counter implementation. The delay is almost the same between the two schemes.

Table I (b) lists the result for hybrid neurons for $P = 16$. Though the impact of LFSRs on the LUT count became smaller than that of the serial neurons, the LFSR implementation consumed 12.1% fewer LUTs for $T = 3840$.

TABLE I
Synthesis results.

(a) Serial neuron

			binary			LFSR		
N	T	#cycle	#FF	#LUT	delay [ns]	#FF	#LUT	delay [ns]
256	120	257	7	7	4.391	7	4	4.358
512	240	513	8	9	4.387	8	4	4.376
1024	480	1025	9	9	4.524	9	5	4.509
2048	960	2049	10	10	4.414	10	5	4.360
4096	1920	4097	11	12	4.519	11	5	4.353
8192	3840	8193	12	16	4.502	12	5	4.353

(b) Parallel-serial hybrid neuron ($P = 16$)

			binary			LFSR		
N	T	#cycle	#FF	#LUT	delay [ns]	#FF	#LUT	delay [ns]
256	120	17	7	31	3.973	8	27	3.995
512	240	33	8	29	3.965	9	27	3.973
1024	480	65	9	29	3.961	10	28	4.106
2048	960	129	10	31	3.998	10	28	4.106
4096	1920	257	11	32	4.110	11	28	4.347
8192	3840	513	12	33	4.376	12	29	4.383

Target: Xilinx Artix-7, Synthesizer: Xilinx Vivado (2016.4)

Considering that the delay of the serial and hybrid neurons is almost the same, the hybrid neurons are more efficient in terms of operations per second per LUT.

V. Conclusion

An idea of implementing BNN neurons using LFSRs has been presented. This helps accommodating more neurons to exploit parallelism in neural inference.

We are working on designing a whole layer or a whole BNN to evaluate the impact of our scheme. Moreover, the latest FPGA chips have various features and auxiliary circuits, which we will exploit to implement smaller neuron modules.

Acknowledgments

Authors would like to express their appreciation to Dr. Hiroyuki Kanbara of ASTEM/RI, Prof. Hiroyuki Tomiyama of Ritsumeikan University, and Mr. Takayuki Nakatani (formerly with Ritsumeikan University) for their discussion and valuable comments. We would also like to thank to the members of Ishiura Lab. of Kwansei Gakuin University.

References

[1] M. Courbariaux, et al.: "Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or −1," Computer Research Repository (Mar. 2016).

[2] H. Nakahara, H. Yonekawa, T. Sasao, H. Iwamoto, and M. Motomura: "A memory-based realization of a binarized deep convolutional neural network," in *Proc. FPT 2016*, pp. 273–276 (Dec. 2016).

[3] K. Ando, et al.: "BRein memory: A single-chip binary/ternary reconfigurable in-memory deep neural network accelerator achieving 1.4 TOPS at 0.6 W," *IEEE J. Solid-State Circuits*, 53-5, pp. 983–994 (Apr. 2018).

[4] R. Sugimoto and N. Ishiura: "Parameter embedding for efficient FPGA implementation of binarized neural networks," in *Proc. SASIMI 2019*, pp. 41-45 (Oct. 2019).