Alpha-gamma Discrimination in BaF_2 Using FPGA-based Feedforward Neural Network

Chenfei Yang, Changqing Feng, Wenhao Dong, Di Jiang, Zhongtao Shen, Shubin Liu, and Qi An

***Abstract*—In this paper, we investigated the hardware implementation of two feedforward neural networks (NNs) using a field-programmable gate array (FPGA), then used the networks for alpha-gamma discrimination in barium fluoride (BaF_2). The BaF_2 detector output was sampled using a 1 GSPS ADC, and then we extracted 6 information of the pulses in FPGA as the input features to the neural networks. The performance of this method turned out very good, the false alarm rate (FAR) of the networks were less than 0.3%. Besides, dead time of the networks were less than 820 ns. Low logic occupancy is also discussed.**

***Index Terms*—Alpha-gamma discrimination, BaF_2 detectors, field-programmable gate array (FPGA), hardware implementation, neural networks (NNs).**

I. Introduction

ARTIFICIAL Neural Networks (ANNs) have been proved capable to map, model and classify nonlinear systems, and are used extensively in recent years. Most of the existing ANN applications are developed as software, in contrast to those designed with hardware devices, e.g. field-programmable gate arrays (FPGAs), which are referred as hardware neural networks (HNNs). However, HNNs can offer appreciable advantages in Speed, Cost and Graceful degradation [1]. As these advantages are especially important in nuclear science, because most experiments in this field have high event rates and large channel numbers, it could be effective to use HNNs in them.

In this work, we implemented feedforward neural networks in a FPGA, and then utilized them for gamma-alpha discrimination in barium fluoride (BaF_2). Since the intense, fast component of the light pulse from BaF_2 crystal was discovered in 1980s [2], BaF_2 scintillators have been widely used in nuclear physics and elementary-particle physics for detecting gamma radiation and light charged particles, especially in medium and high energy physics experiments. Actually, the radio luminescence spectrum of BaF_2 contains two component: a fast component with emission time of 0.6 ns and a slow component with that of 620 ns, and the fast/slow component ratio depends on the particle nature, being larger for gamma-rays and smaller for heavy particles [3]. This feature offers the possibility for one to discriminate alpha-particles and gamma-rays in BaF_2. Besides, BaF_2 crystals always contain some radium impurities, which lead to a typical background mainly constituted by alpha-particles and gamma-rays [4], so we can obtain both alpha and gamma induced events with BaF_2 crystals alone. These properties make BaF_2 a perfect material for studies of alpha-gamma discrimination. On the other hand, the discrimination technology makes works like alpha background elimination possible in BaF_2.

Previous works on particle discrimination can be separated into two main types. Former works were mainly based on analog methods, which needed complex circuits and always had long dead time [5], [6]. Recent works tended to use digital algorithms, most of them were executed offline, and only 1 or 2 characters of the pulses can be considered at same time [7], [8]. The neural networks in our work achieved real-time discrimination, in which we extracted 6 characters from the BaF_2 pulses after 1 GSPS sampling using chip ADC12D1000, and incorporated them all into consideration.

We designed and trained two 6-input, 2-output feedforward neural networks in MATLAB using back-propagation (BP) algorithm, one network with linear excitation function and had 1 layer, and the other one with sigmoid excitation function and had 3 layers, then we implemented them in a Spartan-6 XC6SLX45-2FGG484 FPGA. To evaluate the performances, we tested the network with BaF_2 background radioactivity and a ^{22}Na gamma-source. The results turned out to be far superior to pervious works. In addition, three important factors, i.e. the dead time of the neural network, the precision of the implementation and the hardware logic occupancy, are also elaborated in this paper.

II. BaF_2 Detector Setup and Signal Processing

A. *BaF_2 Detector Introduction and Setup*

Because of the light quenching effect in BaF_2 crystal, the fast/slow component ratio is larger for gamma-induced events than alpha-induced events as mentioned above. Fig. 1 presents two typical waveforms from BaF_2, as can be seen, the alpha induced pulse consists mostly of slow component, and the gamma induced pulse has a large fast component and a small slow component. Exploiting this feature, we extracted 6 pulse characters that contributed to pulse discrimination, which will be detailed as following.

Manuscript received June 24, 2016; revised March 23, 2017.

This work was supported by the National Natural Science Funds of China under Grant 11205154.

The authors are with the State Key Laboratory of Particle Detection and Electronics, and Department of Modern Physics, University of Science and Technology of China, Hefei, Anhui 230026 China (email: fengcq@ustc.edu.cn, feifei55@mail.ustc.edu.cn).

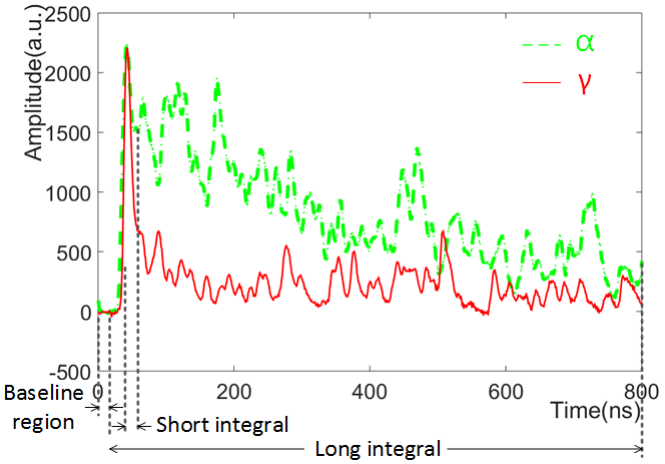

Fig. 1. Typical signal waveforms from BaF_2 crystal sampled at 2 GSPS. The green dotted waveform corresponds to alpha-particle, and the red solid one is gamma-ray. The regions for baseline, long integral and short integral calculations are also indicated.

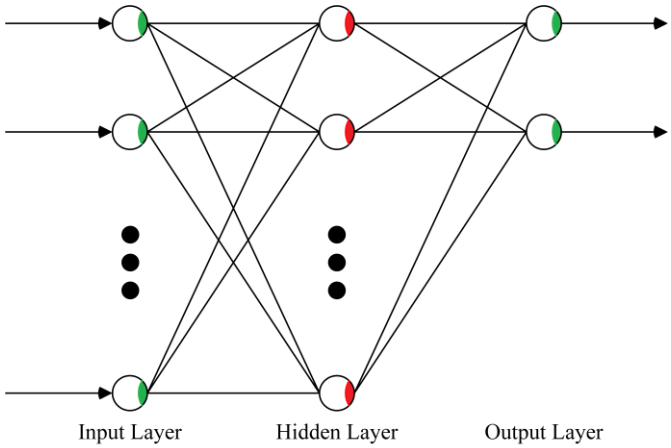

Fig. 2. Architecture of the tan-sig network with 6 input features. The circles with green area indicate neurons using linear excitation function, and those with red area indicate neurons using tan-sig excitation function.

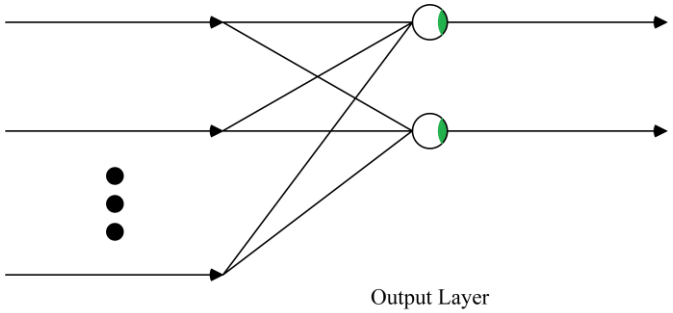

Fig. 3. Architecture of the linear network. The circles with green area indicate neurons using linear excitation function.

B. Waveform Sampling and Digital Processing

The BaF_2 detector output was sampled by a 1 GSPS, 12-bit ADC (ADC12D1000) sampling module. With the sampling depth of 1024, about 1 μs pulse waveform was sampled for each event.

ADC12D1000 was set to work in Demux Mode, in which the ADC provided data at half the sampling clock rate on 2 sets of 12-bit differential data bus. It means that the data rate from ADC to FPGA IO on each port was 500 Mb/s. Then we used the Spartan-6 FPGA IO resources ISERDES2 to make serial-to-parallel conversion with SerDes ratio of 1:4 [9], four 12-bit data were converted to a 48-bit datum and then wrote to a FIFO at one quarter of the IO clock rate (125 MHz). The FIFO was read at 50 MHz, which was also the main clock rate of the FPGA. 1024 samples (about 1 μs pulse waveform) would be read out from the FIFO when the ADC data was over threshold, which was judged by a discriminator in FPGA.

After each pulse being read out from the FIFO, we extracted 6 characters of it as the input features of the network which would be designed later, using FPGA-based digital algorithms. The characters include:

- Pedestal: The average amplitude of first 16 samples, calculated by additions and shift. It can remove the influence of DC bias. Expected range: -2 to +2 ADC units (a.u.).
- Peak: The amplitude of the peak point. Represents energy information of each pulse. Expected range: 80 to 2500 a.u.
- Discrimination gradient: The amplitude difference between the peak point and the 10th sample after peak. Represents the amplitude of the slow component. Expected range: 0 to 600 a.u.
- Over-threshold number: Number of samples over a certain amplitude threshold, which was set to 10 a.u. in this work. Events with rather small over-threshold numbers can be justified as one-photon events. Expected range: 0 to 850.
- Long integral: Sum of the whole pulse samples' amplitude. Represents the energy information of the whole pulse. Expected range: 0 to 1.1e5 a.u.
- Short integral: Sum of a certain region around the peak point, and in this work, the region is from 5th sample before peak to 10th sample after peak. Represents the energy information of the fast component. Expected range: 0 to 1.3e4 a.u.

Those characters would be done extracted in FPGA simultaneously with reading ADC data out from the FIFO, then a bad pulse elimination would be executed. We used the pulse characters to judge whether or not any errors had occurred during sampling, and bad pulses were abandoned. Pulses with characters over expected ranges were also abandoned.

III. Network Design and Implementation

A. Network Design and Training

1) Neuron structure

Neurons are the basic elements of a neural network. And in a feedforward neural network, neurons do multiplications and additions with the inputs from the previous layer neurons and their own weights and biases, then calculate the results with an excitation function, and lastly, send the output to the next layer neurons. The excitation function can be linear or non-linear, to compare the performance in discrimination of these two types, we designed two feedforward neural networks, one (linear network) used all linear excitation functions, as in (1), and the other one (tan-sig network) used tan-sig excitation functions, as

in (2), for neurons of a certain layer. The excitation functions are listed as follows:

- Linear excitation function

$$f(x) = x; \tag{1}$$

- Tan-sig excitation function

$$f(x) = \frac{2}{1+e^{-2x}} 1. \tag{2}$$

2) Tan-sig Network Design

The tan-sig network was designed to have 3 layers: input layer, hidden layer and output layer as shown in Fig. 2. Neurons in the input layer and output layer used linear excitation functions, and neurons in the hidden layer used the tan-sig excitation function.

After digital processing the 6 characters were sent to the neural networks as the input features, i.e. the network should have 6 neurons in the input layer. The hidden layer also had 6 neurons. And in the output layer, each neuron could indicate two states (0 or 1), just like a binary bit. To indicate two kinds of discrimination result, i.e. gamma and alpha, at least 1 neuron was needed in the output layer. However, to save possibilities for future improvements, such as identifying noises and bad pulses, we reserved one more neurons, which made the output layer had 2 neurons.

3) Linear Network Design

Different from the tan-sig network, neurons all used linear excitation functions in the linear network, so it just needed one output layer with two neurons to complete all the calculations. The architecture of the linear network is shown in Fig. 3.

4) Principle of network training

After design of the networks, we trained them in MATLAB using back-propagation (BP) algorithm [10], which had succeed in pattern recognition, self-adaptive filtering, deep learning networks and many other works. BP algorithm needed a training set with input features and corresponding targets, and its training principle was to continuously adjust the weights and biases of the network, making the network outputs keep approaching the corresponding training set targets. The differences between the network outputs and the training set targets was calculated by a cost function as in

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 \tag{3}$$

where x is the input feature, y is the corresponding target, θ is the number of adjustments, and h_θ is a function equivalent to the whole network.

A validation set with input features and corresponding targets was also applied. After each adjustment of weights and biases, we compared the cost function of the validation set with the cost function of it before current adjustment to avoid overfitting. And finally, the training process would be completed by obtaining a set of weights and biases which made the cost function of the training set the least, and without overfitting.

5) Generation of Training and Validation Sets

The training and validation sets are sets of events which types (alpha or gamma) were already known. To generate them, we applied a pulse shape discrimination (PSD) algorithm called pulse gradient analysis (PGA) [11]. It was an algorithm using two pulse characters (long integral and discrimination gradient) as mentioned above. The result of PGA algorithm is presented in Fig. 4, the discrimination threshold line was determined by a rough method. It can be seen that some of the events are close to the discrimination threshold line, which would be identified as different types when the line moving, so these events were not suitable to be used as training set or validation set. We selected events far from the discrimination threshold line (typical events), on the other sides of the selection lines, 50,000 events as the training sets, and 32,000 other events as the validation sets. It's remarkable that the propagations of alpha and gamma events among the training and validation sets should be equal to the propagations of them among the whole events, which would avoid the network overweighting on any event types. Then we used target outputs of (0, 0) to indicate typical gamma events, and (1, 1) to indicate typical alpha events. With the same input features and corresponding targets for the two network outputs, they would be equivalent after training.

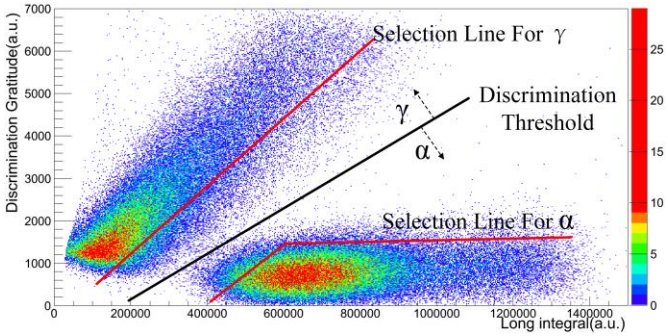

Fig. 4. Result of PGA algorithms, and selection of typical alpha and gamma events. Events above the discrimination threshold line were identified as gammas by PGA, and events below the line as alphas. Typical events were selected on the other side of the selection line far from the discrimination line.

Using the method above, the training and validation sets were generated, and then we finished training the feedforward neural networks.

B. Network Implementation

1) Basic Frame

As described above, the network processing was constituted by multiplications, additions and a tan-sig excitation function (just the tan-sig network). Only the implementation of the tan-sig network is introduced in this paper, because the implementation of the linear network can be seen part of the tan-sig network implementation.

Generally, the multiplications and additions were achieved with multipliers and adders, and the tan-sig excitation function was achieved by using a ROM as look up table (LUT) in FPGA.

If we used multipliers and adders that match the maximum number of the calculations of a single layer and ran all of them in parallel, which could be referred as the Whole Parallel Pattern, the whole network processing could complete in several clock cycles. But also, it meant very large logic occupancy. To lessen the logic occupancy, we could run the network in a Whole or Part Time-sharing Pattern, which was also referred as multiplexing, that could reduce the amount of multipliers and adders. The Whole Time-sharing Pattern means running all the calculations in time-sharing pattern with only one multiplier and one adder instance (one Neuron Implementation). The Part Time-sharing Pattern means running the calculations of each neuron in time-sharing pattern with one

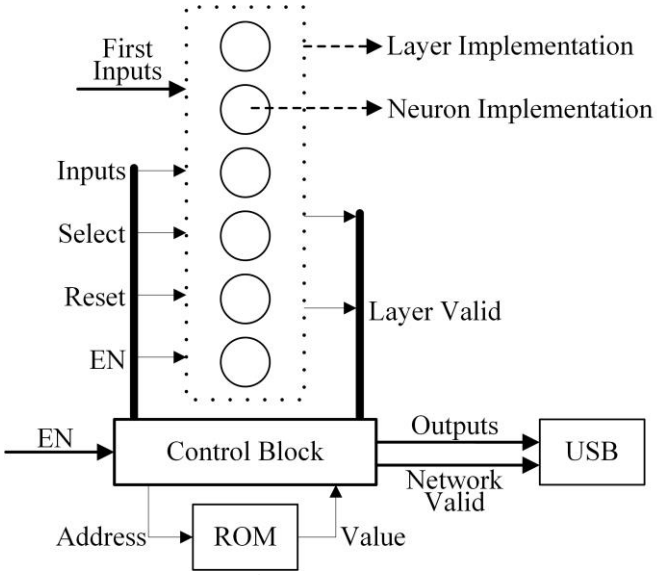

Fig. 5. Schematic diagram of the Part Time-sharing Pattern controlling in the network implementation.

TABLE I
CALCULATED PARAMETERS OF DIFFERENT WORKING PATTERNS

Working Pattern	Multipliers Amount	Dead Time (Cycles)
Whole Parallel	36	25
Part Time-sharing	6	36
Whole Time-sharing	1	107

Neuron Implementation, and different neurons of each layer in parallel with several Neuron Implementations. The number of Neuron Implementations should match the maximum number of neurons in a single layer. Both the Time-sharing Patterns would cause less logic occupancy but with longer dead time [12]. As another choice, we could reduce the word lengths of the multipliers and adders, however it would on the other hand reduce the output precision. From here we see that the dead time, the precision, and the logic occupancy were constrained by each other.

2) Network Working Pattern and Dead Time

To choose the network working pattern, we calculated and compared the logic occupancies and dead time of the Whole Parallel Pattern, the Whole Time-sharing Pattern and the Part Time-sharing Pattern as shown in Table I, where the logic occupancies are represented by the amount of the multipliers needed.

After considering the logic occupancy, the dead time, and also the logic resource of the Spartan-6 FPGA we used, we decided to run the network in Part Time-sharing Pattern. Specifically, it meant computing different layers using same Layer Implementation at different time, and it was arranged by a control block as shown in Fig. 5. And the maximum number of neurons of one single layer was 6 (the input layer and the hidden layer), so the Layer Implementation should contain 6 Neuron Implementations. The 6 Neuron Implementations were same in structure, only with different weight and bias values.

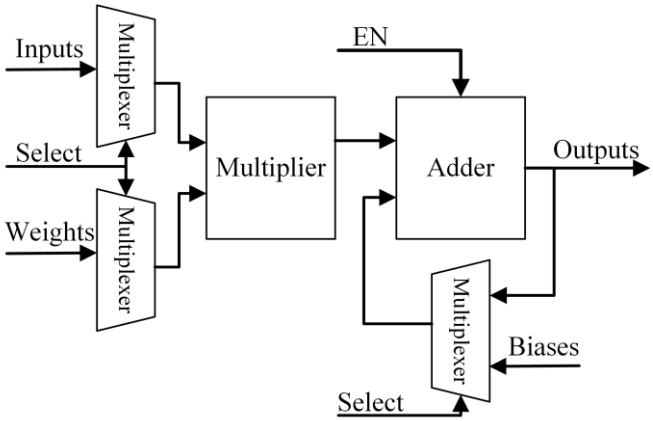

Fig. 6. Architecture of the Neuron Implementation.

We designed the Neuron Implementations using XILINX System Generator, and the software could generate the schematics into hardware description languages (HDLs) [13], [14]. The architecture of the Neuron Implementation is presented in Fig. 6. Use a neuron in the hidden layer for example, it had 6 inputs from the previous layer, so 6 multiplications need to be done. Multiplexers were employed for selecting the inputs to the multiplier, and an adder was used for adding the multiplier outputs and biases.

The number of multiplications in each neuron was depended on the number of inputs from the previous layer. Accordingly, neurons in the input layer had 1 multiplication, and neurons in the hidden and output layers had 6 multiplications. And the multipliers had same latency of 3 clock cycles. So the input layer neurons needed at least 5 clock cycles for computing (1 for multiplication, 3 as latency and 1 for bias addition), i.e. the dead time of the neuron, and neurons of other layers needed at least 10 cycles (6 for multiplication, 3 as latency and 1 for bias addition). Considering that neurons in one layer computed in parallel, and layers computed in sequence, the dead time of the whole network should be more than 25 clock cycles theoretically, i.e. 500 ns with our FPGA ran at 50 MHz. But actually, after implementation, there were redundant time in the control block, and added the whole dead time to 720 ns (36 clock cycles). For the linear network, the actual whole dead time was 260 ns, compare to the theoretical dead time of 100 ns. However, 720 ns dead time was fully acceptable considering the 2000 ns dead time of sampling for each pulse.

3) Network Precision and Logic Occupancy

The output word lengths of every part in the network would affect to the output precision. The selection of the multipliers and adders output word length should take factors as follows into account: input numbers of neurons, weights and biases precision, and of course, requirements for hardware resources. The output word lengths of all multipliers and adders were set to 40 bits (1 bit for sign + 7 bits for whole part + 32 bits for fractional part). The FPGA we used (XC6SLX45) only has 58 DSP slices (DSP48A1), and most had been used for digital processing of the pulses, so all the multipliers and adders were implemented using fabrics. 1932 Slice LUTs were occupied for multipliers and adders in the tan-sig network, and 557 in the linear network.

The ROM in the tan-sig network was used as LUT to accomplish the tan-sig excitation function, its precision was

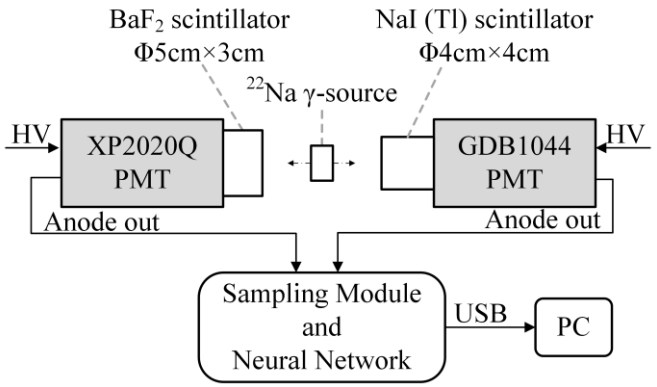

Fig. 7. Schematic diagram of ^{22}Na gamma-source experiment.

decided by the memory depth and word length [12], and at last the ROM was designed with memory depth of 16,384 cells (14 bits address) and word length of 16 bits (1 bit for sign + 3 bits for whole part + 12bits for fractional part), and only capable for input from -8 to +8. Input greater than +8 would led to a +1 output, and input less than -8 would led to a -1 output, by which only brought an error less than 3e-7 in the tan-sig function calculation. The FPGA was also lack of RAM16Bs because the sampling process used most of them, so the ROM was also implemented using Slice LUTs, and 2366 Slice LUTs were occupied.

In summary, the tan-sig network occupied 4298 Slice LUTs (27,288 available in XC6SLX45), and the linear network occupied 557 Slice LUTs.

IV. Experimental and Results

A. Experimental Setup

To evaluate the network performance in discrimination, we performed two experiments, one with the natural background of BaF_2 crystal, the other one with the radioactivity of a ^{22}Na gamma-source. In the first experiment we just connected the BaF_2 detector output to the sampling module, and the event rate was about 130 Hz with the trigger threshold set to 10 mV. And in the second one, a NaI (Tl) detector was set opposite to the BaF_2 detector for coincidence. Between the two detectors we placed a collimated ^{22}Na gamma-source. And owing to the two-photon radiation of ^{22}Na, coincident events of these two detectors should be gamma events except for few random coincidence alpha events. Both detector outputs were sampled by the sampling module. In this experiment the event rate of BaF_2 was about 4.07 kHz with 10 mV trigger threshold, and the event rate of detected coincident event was about 425 Hz. Fig. 7 shows the experimental setup of the second experiment. Because that in this work two network outputs were equivalent, so we will just analyze the first one of the outputs, as OutputA.

B. Results with Tan-sig Network

1) Results of BaF_2 Natural Background

The distribution of tan-sig network OutputA for 200,000 BaF_2 natural background events is shown in Fig. 8. Events were separated into 2 peaks of alpha and gamma events. We fitted them with Gaussian functions and obtained the mean values and FWHMs of both peaks. The discrimination threshold was set as average value of two mean values (0.4989), that is to say

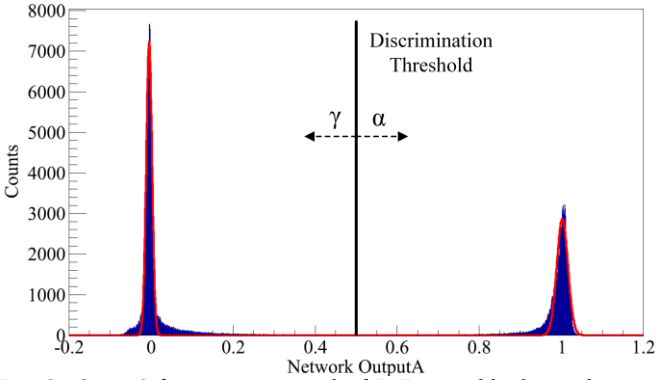

Fig. 8. OutputA from tan-sig network of BaF_2 natural background events. The red line is the Gaussian fitting result.

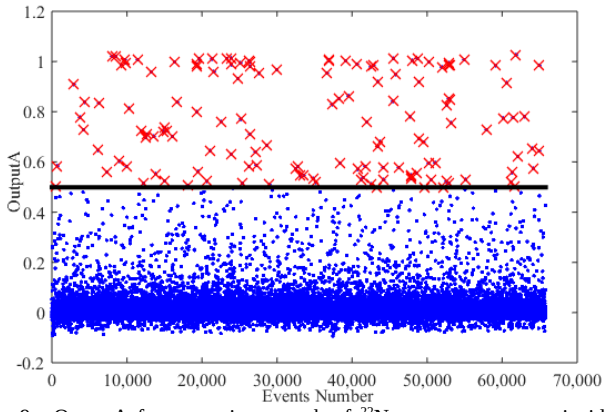

Fig. 9. OutputA from tan-sig network of ^{22}Na gamma-source coincident events. The black line is the discrimination threshold.

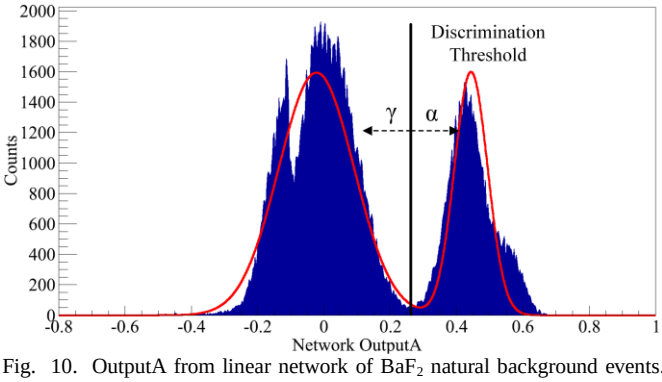

Fig. 10. OutputA from linear network of BaF_2 natural background events. The red line is the Gaussian fitting result.

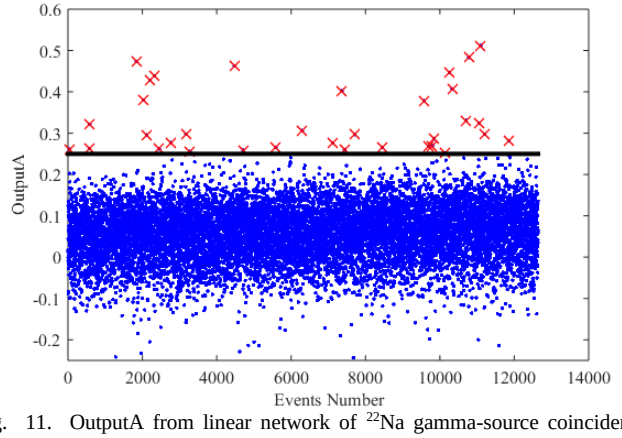

Fig. 11. OutputA from linear network of ^{22}Na gamma-source coincident events. The black line is the discrimination threshold.

events with OutputA larger than the threshold were identified as alphas, and events with OutputA smaller than the threshold were identified as gammas. To evaluate the separation of two peaks, figure-to-merit (FOM) was calculated as in

$$FOM = \frac{Mean_{\alpha} - Mean_{\gamma}}{FWHM_{\alpha} + FWHM_{\gamma}}. \quad (3)$$

Larger FOM means that alpha peak and gamma peak are more separated. FOM of tan-sig network OutputA is 14.52, and as a comparison, the FOM of the software-based tan-sig network OutputA before implementation was 14.09. The implemented network FOM is even higher than the software-based network FOM.

2) Results of ^{22}Na gamma-source radioactivity

We defined the coincident events as when BaF_2 detector had a pulse bigger than 10 mV, the NaI (Tl) detector also gave out a pulse bigger than 10 mV in the same 100 ns time range. The tan-sig network OutputA for 65,808 coincident events which should had been all identified as gammas. Whereas there were 138 events mistakenly identified as alphas are shown by red crosses in Fig. 9. False alarm rate (FAR) was defined as the ratio of mistakenly discriminated events, and FAR for tan-sig network was 0.0021 (0.21%). And obviously, lower FAR means better discrimination performance.

C. Results with Linear Network

1) Results of BaF_2 Natural Background

The distribution of linear network OutputA for 200,000 BaF_2 natural background events is shown in Fig. 10. The discrimination threshold was set at the valley between two peaks (0.2621). The steep shape in the gamma peak was because linear system did not converge well. FOM of implemented linear network OutputA was 0.76, and the FOM of the software-based linear network OutputA was 0.85. The implemented network FOM is a little smaller than the software-based network FOM.

2) Results of ^{22}Na gamma-source radioactivity

The linear network OutputA for 12,638 coincident events are presented in Fig. 11. There were 34 of them greater than the discrimination threshold and being identified as alphas. The FAR of linear network was 0.0027 (0.27%).

V. Discussions and Conclusion

In this work we designed and implemented two feedforward networks in FPGA for alpha-gamma discrimination in BaF_2, one used tan-sig excitation function and the other one only used linear excitation function. We used two parameters to describe their performance, FOM and FAR, and compared them in Table II. PGA, the algorithm we used to generate the training sets of the networks, was also compared. Tan-sig network FOM and FAR were both far superior to PGA, the advantage of comprehensive consideration of 6 pulse characters instead of 2 was well proved. FOM of linear network turned out to be the worst, this shows that the linear network was not suitable for such discrimination.

On the other hand, we can also evaluate the precision of implementation by the performance difference between the software-based (after training, before implementation) network and the implemented network. As an example, FOM of the software-based tan-sig network was 14.09, which is even slightly worse than the implemented tan-sig network. This proves that the implementation was accomplished very well.

TABLE II
Performance Comparison of Different Methods

Method	FOM	FAR
Tan-sig network	14.52	0.21%
Linear network	0.76	0.27%
PGA	1.22	1.32%

Besides, we have balanced the performance of discrimination, the logic occupancy and the dead time. Still take tan-sig network as an example, logic occupancy of 4298 Slice LUTs would be provided easily, and with the 720 ns dead time (36 clock cycles in FPGA), the network would be capable for very high event rate experiments.

It's also possible to improve this network. If the most amount of neurons in one layer increase, such as changing the input layer to 8 neurons, the whole logic occupancy and dead time would increase linearly with the neuron amount. And if we increase layer number of the network, the dead-time would increase linearly, whereas the logic occupancy would almost not change. Furthermore, if we set more neurons in the output layer, the network would be able to discriminate exponential number of input patterns.

With this network, works such as elimination of alpha background in BaF_2 could be done in real-time with little cost. And refer to this work, except for alpha-gamma discrimination, feedforward networks could be used for many more purposes in particle detection.

References

[1] J. Misra and I. Saha, "Artificial neural networks in hardware: A survey of two decades of progress," *Neurocomputing*, vol. 74(1), pp. 239-255. 2010.

[2] M. Laval, M. Moszyński, R. Allemand, et al., "Barium fluoride—Inorganic scintillator for subnanosecond timing," *Nuclear Instruments and Methods in Physics Research*, vol. 206(1), pp. 169-176. 1983.

[3] E. D. Filippo, G. Lanzano, A. Pagano, et al., "A study of light quenching in BaF2 crystals for heavy ions at intermediate energies," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 342, pp. 527-533. 1994.

[4] K. Wisshak and F. Käppeler, "Large barium fluoride detectors," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 227, pp. 91-96. 1984.

[5] M. L. Roush, M. A. Wilson and W. F. Homyak, "Pulse shape discrimination," *Nuclear Instruments and Methods*, vol. 31(1), pp. 112-124. 1964.

[6] R. Aryaeinejad, E. L. Reber and D. F. Spencer, "Development of a handled device for simultaneous monitoring of fast neutrons and gamma rays," *IEEE Transactions on Nuclear Science*, vol. 49(4), pp. 1909-1913. 2002.

[7] R. Aryaeinejad, J. K. Hartwell and D. F. Spencer, "Comparison between digital and analog pulse shape discrimination techniques for neutron and gamma ray separation," *Nuclear Science Symposium Conference Record*, 2005 IEEE, vol. 1, pp. 500-504. 2005.

[8] K. A. A. Gamage, M. J. Joyce and N. P. Hawkes, “A comparison of four different digital algorithms for pulse-shape discrimination in fast scintillators,” *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 642(1), pp. 78-83.2011.

[9] XILINX INC. *Spartan-6 FPGA SelectIO Resources User Guide*. (2015) [Online]. Available: https://www.xilinx.com/support/documentation /user_guides/ug381.pdf, Accessed on: Nov. 28, 2016.

[10] S. S. Haykin, “Multilayer Perceptrons,” in *Neural networks: a comprehensive foundation*, 2nd ed. Tsinghua University Press, 2001.

[11] B. D’Mellow, M. D. Aspinall, R. O. Mackin, et al., “Digital discrimination of neutrons and γ-rays in liquid scintillators using pulse gradient analysis,” *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 578(1), pp. 191-197. 2007.

[12] S. Himavathi, D. Anitha and A. Muthuramalingam, “Feedforward neural network implementation in FPGA using layer multiplexing for effective resource utilization,” *IEEE Transactions on Neural Networks*, vol. 18(3), pp. 880-888. 2007.

[13] XILINX INC. *System Generator for DSP User Guide*. (2012) [Online]. Available: https://china.xilinx.com/support/documentation/sw_manuals /xilinx14_7/sysgen_user.pdf, Accessed on: Nov. 30, 2016.

[14] M. Bahoura and H. Ezzaidi, “FPGA-implementation of a sequential adaptive noise canceller using Xilinx system generator,” in IEEE 2009 International Conference on Microelectronics, pp. 213-216.