# A Twofold Lookup Table Architecture for Efficient Approximation of Activation Functions

Yusheng Xie, Alex Noel Joseph Raj, Zhendong Hu, Shaohaohan Huang, Zhun Fan, *Senior Member, IEEE*, and Miroslav Joler, *Senior Member, IEEE*

***Abstract*— In this article, we propose a novel approach to reduce hardware resource consumption when neural networks (NNs) are deployed on field-programmable gate array (FPGA) boards. Rather than using a classical approach with lookup tables (LUTs) to approximate the activation functions of an NN, the proposed solution is based on a twofold LUT (t-LUT) architecture, which comprises an error-LUT (e-LUT) and a data-LUT (d-LUT), in order to achieve high precision and speed as well as low hardware resource consumption. The efficiency of the proposed approach was tested against multiple earlier approaches. Our solution showed that the compressibility of the previously referenced works, which were based on single LUTs, could be improved by up to 94.44% and those that were based on a range addressable LUT (RALUT) by up to 6.35% in the examined case of a hyperbolic tangent (tanh) activation function. Moreover, when RALUT and our architecture were combined, it improved the compressibility of the RALUT-based result by up to additional 10.21% for a tanh activation function. The designed architecture had an initial latency of 39.721 ns, when tested with a 50-MHz clock, to simultaneously retrieve data from the d-LUT and t-LUTs.**

***Index Terms*— Activation functions, field-programmable gate array (FPGA), twofold lookup table (t-LUT).**

## I. Introduction

With the rapid development of artificial intelligence, neural networks (NN) play a vital role in many fields, such as computer vision, natural language processing, and automatic driving. Generally speaking, most of these NNs are implemented using a combination of a CPU and graphics processing unit (GPU), which results in high costs and high power consumption. Recently, NNs have been implemented on a field-programmable gate array (FPGA) with low-cost and low-power architectures. With that being done, a computing activation function of an NN remains a crucial process.

Due to the hardware limitations in FPGA, activation functions are all based on the approximation rather than being calculated on the FPGA. Researchers implement the activation functions by the following three typical approaches: 1) piecewise linear approximation and piecewise nonlinear approximation [1], [2]; 2) lookup tables (LUTs) [3]; and 3) hybrid methods using a combination of the first two methods [4]–[7]. Among these, LUT is the fastest method when compared with the other different hardware implementations as it requires fewer computations to compute the activation function.

Recently, LUT methods that are characterized by implementations with a smaller area, shorter latency, and higher accuracy were presented. These methods either combined the LUTs with linear interpolation procedure [8] or used range addressable LUT (RALUT) [9]–[12] mapping unique outputs to an address range. However, with a high precision [9]–[11], RALUT also needed a number of comparators, which then involves a high area complexity. In [12], a customized logic was used to design an address range decoder, where a simplification of a Boolean expression for necessary comparisons was arduous. Some other proposed methods enabled the use of the same LUTs [13] to reduce the memory space or employ compressibility within the coding process to optimize the table size [14]. In [15], Taylor series expansion [16]–[19] was used to decompose and compress the signal and the middle value was used in the tables as the reference point to represent the consecutive values in the original LUT, leading to differences that are either positive or negative, while no precondition was set for the compression. Linear LUTs present a simple architecture where the keywords are directly mapped to an address, but this results in a heavy hardware usage. Reduction in hardware usage can be achieved by finding a suitable transformation function that maps the keywords to an address, but, again, designing such an architecture is not trivial.

In this article, we propose an LUT architecture that is based on a direct mapping method, the minimum value (rather than the middle value) and two LUTs that reduce the complexity of storage and computation, as it will be explained. Although the transformation of double-precision floating-point values to fixed-point values (Fig. 1) results in many duplicate values, we circumvent that by forming bands of neighboring values grouped together where from each band a minimum value is chosen to be stored in an LUT referred to as the data-LUT (d-LUT). Next, we calculate the difference between the original value and the minimum value stored in the d-LUT and store the error value in another LUT called the error-LUT (e-LUT). Thus, the original LUT values are obtained by adding the minimum value and the error that are

Manuscript received February 23, 2020; revised June 1, 2020 and July 20, 2020; accepted August 4, 2020. This work was supported by the Scientific Research Grant of Shantou University, China under Grant NTF17016. *(Corresponding author: Alex Noel Joseph Raj.)*

Yusheng Xie, Alex Noel Joseph Raj, Zhendong Hu, Shaohaohan Huang, and Zhun Fan are with the Department of Electronic Engineering, Shantou University, Shantou 515063, China (e-mail: jalexnoel@stu.edu.cn).

Miroslav Joler is with the Department of Computer Engineering, University of Rijeka, Rijeka 51000, Croatia.

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

Digital Object Identifier 10.1109/TVLSI.2020.3015391

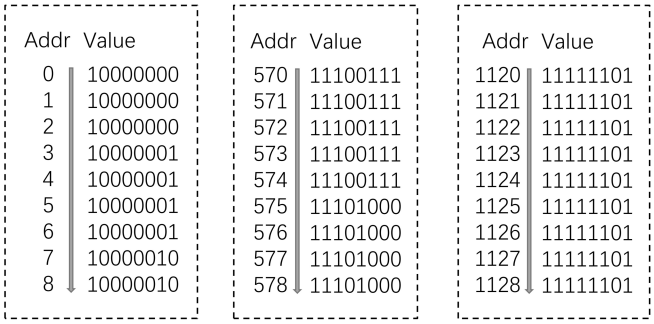



Fig. 1. Excerpts from sigmoid function LUT. Many duplicates exist.

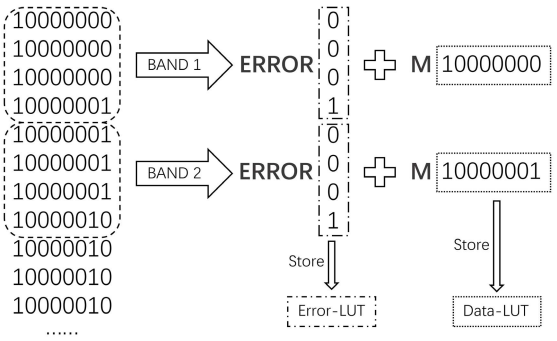



Fig. 2. Abstract of t-LUT architecture.

simultaneously stored in the d-LUT and e-LUT. By choosing the minimum value to represent a band of consecutive values, we ensure that the differences are all only positive values, which simplifies the computation by not having to work with the sign bit. Moreover, we defined a precondition, which enabled us to compute the compressibility mathematically and present the optimal compressibility for various widths of LUTs and band sizes. Our experiments with different activation functions, such as sigmoid, tanh, softplus, and more, proved that the activation functions approximated based on the proposed twofold LUT (t-LUT) structure require only half of the total number of bits compared to the original LUT.

The rest of this article is organized as follows. Section II explains the t-LUT architecture mathematically. Subsequently, the procedure of designing our LUT is shown in Section III. The implementation details and the hardware utilization are presented in Section IV. In Section V, we discuss the limitations and the advantages of the proposed method and provide comparisons with other designs. Finally, Section VI concludes this article.

## II. Principle of t-LUT

In this section, the idea behind the proposed t-LUT is described, followed by the theoretical discussion of compressibility between the original LUT and t-LUT.

### A. t-LUT Architecture

When we approximate an activation function using an LUT (referred to as the single LUT structure) in an FPGA, the fixed point representation is usually a preferred choice to represent the data. Unfortunately, due to the quantization effects, the approach presents many duplicate values, which causes a huge loss of valuable resources (see Fig. 1). A higher compressibility can be achieved by widening the sampling intervals, but this results in an increased error.

To design the t-LUT architecture, we assume that the difference between two consecutive values in the original LUT should be either 0 or $\pm$ 1 (we will later discuss in Section V the situation where data do not meet this precondition). We form a band by grouping $B$ neighboring data together (here, we limit $B = 2^k (k \in Z^+)$ for computational simplicity) and choose the minimum value $M = \min(v_1, v_2, v_3, \ldots, v_i, \ldots, v_B), v_i \in$ band, $i = 1, 2, 3, \ldots, B$ as the representation value of this band. We store all $M$'s in a d-LUT and employ another LUT referred to as e-LUT to store all the differences between original LUT values and $M$'s. While retrieving them, values $M$'s and error values are simultaneously read from d-LUT and e-LUT, followed by the sign-extension added in addition to $M$'s. Thus, the original values are intact while extracting from the t-LUT. Fig. 2 shows the proposed t-LUT architecture.

### B. Bit Width Requirements of t-LUT and Compressibility

Assume that we have built an LUT with size $T_0$ given by

$$T_0 = W_0 \times D_0 \,(\text{bits}) \tag{1}$$

where $W_0$ is the width in bits and $D_0$ is the depth of the LUT ($D_0$ depends on the designed techniques). Let $B$ be the band formed with $2^k (k \in Z^+)$ values and $E$ be the number of errors in the band. After grouping, the depth of the d-LUT is determined by

$$D_d = \text{ceil}\left(\frac{D_0}{B}\right) \tag{2}$$

where ceil$(\cdot)$ means round up to an integer and the width of d-LUT $W_d$ remains the same as $W_0$. The depth of the e-LUT is determined by

$$D_e = D_0. \tag{3}$$

In theory, the band size $B \geq E$, and then, the width of e-LUT $W_e$ can be determined by

$$W_e = \text{ceil}(\log_2 E) \leq \log_2 B. \tag{4}$$

For a given $W_e$ and $W_0$, the compressibility $C$ is defined as

$$C = 1 - \frac{T_e + T_d}{T_o} = 1 - \frac{W_e \times D_0 + W_0 \times \text{ceil}\left(\frac{D_0}{B}\right)}{W_0 \times D_0} \tag{5}$$

where

$$T_e = W_e \times D_e = W_e \times D_0 \,(\text{bits}) \tag{6}$$

and

$$T_d = W_d \times D_d = W_0 \times \text{ceil}\left(\frac{D_0}{B}\right) (\text{bits}) \tag{7}$$

are the size of e-LUT and d-LUT, respectively.

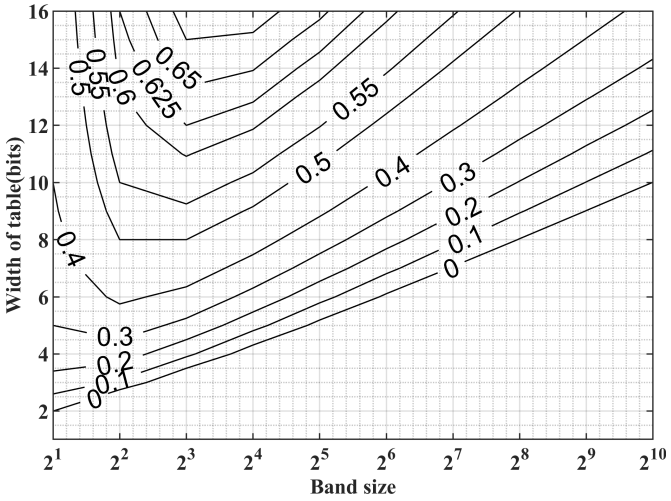



Fig. 3. Relationship between $B$, $W_0$, and $C$ as depicted in (9). As $B$ increases, $C$ will reach a maximum and then decrease. $C \leq 0$ refers to an uncompressed case.

When $D_0 \gg B$, (5) can be approximated as

$$C \approx 1 - \frac{W_e \times D_0 + W_0 \times \frac{D_0}{B}}{W_0 \times D_0} = 1 - \frac{W_e + \frac{W_0}{B}}{W_0} \tag{8}$$

where the compressibility $C$ is independent of the depth of table $D_0$.

Since $B \geq E$, for the worst case scenario, we substitute $W_e = \log_2 B$ to find the relationship between the band size $B$, width of LUT $W_0$, and compressibility $C$. Thus, (8) can be approximated as

$$C \approx 1 - \frac{\log_2 B + \frac{W_0}{B}}{W_0}. \tag{9}$$

Fig. 3 shows the relationship between $B$, $W_0$, and $C$ for the worst case. It can be inferred that for a fixed $W_0$, as $B$ is increasing, $C$ will reach a maximum. Also, irrespective of $B$, larger the size of $W_0$, the better the compressibility performance will be.

Once a proper band size is fixed, we can further improve the compressibility by finding a smaller $W_e = \text{ceil}(\log_2 E)$ (if it exists). For most arbitrary LUTs, the number of continuous values is nonlinear and nonmonotonic. Thus, although it is a rather complex task to design a generic function that will determine $E$, the actual $E$ can be computed through an exhaustive method (see Appendix A). In general, if $\text{ceil}(\log_2 E) < \log_2 B$, the compressibility will be greater.

## III. Design of t-LUT

In this section, we will demonstrate the process of designing a t-LUT for a sigmoid function.

### A. Simplification of the Activation Function

Due to the symmetry of the sigmoid function along the zero, only a portion of the whole function is required to be stored in the LUT. Specifically, if the positive values are computed,

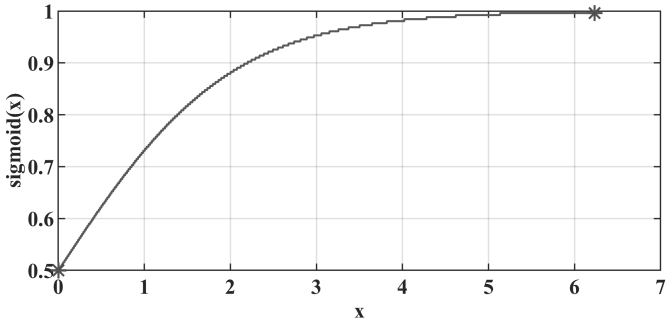



Fig. 4. Right half of a sigmoid function with LT = 0 and HT = 6.234375.

the corresponding output can be obtained by $1-\text{sigmoid}(-x)$, where $x$ is the corresponding negative input. Thus, the sigmoid function $F(x)$ can be represented as

$$F(x) = \begin{cases} \text{sigmoid}(x), & x \geq 0 \\ 1 - \text{sigmoid}(-x), & x < 0 \end{cases} \tag{10}$$

where

$$\text{sigmoid}(x) = \frac{1}{1+e^{-x}}, \quad x \in R.$$

Moreover, once the input exceeds the threshold, the sigmoid function saturates and the output can be approximated to 1. Consequently, for a symmetrical and convergent function, only an output equivalent to $0 \leq x \leq$ Threshold is required to be stored in LUT (see Fig. 4).

Since the LUT output data precision is identical to the NN weights precision, the output data may have a wider range than the sigmoid function. In other words, considering that the NN weights have $m$ integer bits and $n$ fractional bits (since the sigmoid function is mapped between 0 and 1), the information between 0 and 1 is required to be stored. Therefore, only $n$ fractional bits are used to represent the sigmoid function and, thus, by storing only $n$ factional bits for each output, we save an $m$-bit space per output.

Accordingly, we propose three general strategies for simplification of the target function $f(x)$.

1) If function $f(x)$ is symmetrical, only one half of the function is required to be stored in the LUT. The remaining function can be obtained by the symmetrical calculation.
2) If function $f(x)$ is convergent to a constant $C$, then $f(x)$ can be approximated based on the high (HT) and low (LT) threshold conditions

$$f(x) = \begin{cases} \text{LUT}(x), & \text{LT} \leq x \leq \text{HT} \\ C, & \text{else} \end{cases} \tag{11}$$

where $\text{LUT}(x)$ means making inquiry from LUT.
3) If the range of function $f(x)$ is much narrower than the range of the whole data structure, we abandon the unused bits in the LUT. For example, for a sigmoid function of range 0–1, we only need to store the fractional bits to represent all the values and, thus, one can abandon the sign bit as well as the integer bits.

TABLE I
ERROR ANALYSIS OF DIFFERENT DATA PRECISIONS

| Fractional Bits of F(x) | Maximum Absolute Error | Average Absolute Error | Mean Squared Error | Size of Table (bits) | Fractional Bits of x | Sampling Interval | Compressibility |
|---|---|---|---|---|---|---|---|
| 4 | 0.031209 | 0.010784 | 2.042852e-04 | 220 | 4 | 0.06250000 | 24.09% |
| 5 | 0.015620 | 0.006027 | 5.729574e-05 | 665 | 5 | 0.03125000 | 55.44% |
| 6 | 0.007812 | 0.003331 | 1.617626e-05 | 1866 | 6 | 0.01562500 | 58.25% |
| 7 | 0.003906 | 0.001815 | 4.475560e-06 | 4970 | 7 | 0.00781250 | 60.64% |
| 8 | 0.001953 | 0.000971 | 1.226473e-06 | 12776 | 8 | 0.00390625 | 62.48% |
| 9 | 0.000977 | 0.000506 | 3.330143e-07 | 15975 | 8 | 0.00390625 | 54.16% |
| 10 | 0.000488 | 0.000250 | 8.295842e-08 | 9760 | 7 | 0.00781250 | 47.50% |
| 11 | 0.000244 | 0.000119 | 1.911037e-08 | 11264 | 7 | 0.00781250 | 42.05% |
| 12 | 0.000122 | 0.000061 | 4.930143e-09 | 12288 | 7 | 0.00781250 | 37.50% |

Based on these conditions, in Section III-B, we discuss the width and depth of t-LUT.

### *B. Data Precision*

Before we design the t-LUT, we first determine the bit width of $f(x)$ and $x$, i.e., how many integer and fractional bits should be used to represent $f(x)$ and $x$, respectively. Since hardware designs are constrained, due to memory limitations, we initially presented our bit width analysis and attainable compressibility for a block memory size of 18k bits. Later, we relaxed these memory limitations (see Table X) and presented the achievable compressibility for different widths and band sizes.

Now, the number of integer bits of $x$ can be easily determined from the range (from LT to HT). In the case of the sigmoid function, it is between 0 and 6. Therefore, the number of integer bits of $x$ should be 4, including the sign bit. Also, as we explained earlier, the sigmoid function requires only the fractional bits to be stored. Therefore, the number of integer bits of $F(x)$ is 0, while we set the number of fractional bits of $F(x)$ and $x$ to be equal, which can make the LUT meet our precondition more easily (more details are discussed in Section V-A). The number of the fractional bits of $F(x)$ and $x$ determines the precision and the sampling interval, respectively. Here

$$\text{sampling interval } = 2^{-j}, \quad j \leq f \tag{12}$$

where $j$ is the number of the fractional bits of $x$ and $f$ is the number of the fractional bits of $F(x)$. Here, we limit $j \leq f$ because when $j > f$, it does not reduce the mean-squared error (MSE) but exponentially increases the necessary hardware, resulting in huge memory utilization. An experimental analysis shown in Appendix B for $f = 9$ and $j = 8, 9, 10, 11, 12$ clearly illustrates it. Referring to Table I, when $f$ increases from 4 to 8, $j$ becomes equal to $f$, presenting more samples with higher precision. Moreover, when $f$ increased from 8 to 12, $j$ is lower than $f$ to meet the memory limitation.

Table I also shows the maximum absolute error, average absolute error, and MSE between the original $F(x)$ values in double precision and the respective approximated values computed for different data widths of LUTs. When $f$ is increased from 4 to 12 bits, the maximum absolute errors and average absolute errors will decrease significantly. However, when $f$ increases from 8 to 12 bits, the memory usage is larger, yet the compressibility does not increase (how the t-LUT is built will be discussed later). Therefore, we establish a tradeoff between the precision, compressibility, and utilization and fix the precision of both $F(x)$ and $x$ to eight fractional bits for the sigmoid case, for which we can consequently build two LUTs—a single LUT structure and the proposed t-LUT structure—and determine the required depth. For a given sampling interval, the depth of the single LUT is given by

$$D_0 = \frac{\text{HT} - \text{LT}}{\text{sampling interval}} + 1 \tag{13}$$

where HT and LT refer to the upper and lower thresholds of the sigmoid function, respectively. We used MATLAB to compute the threshold values LT and HT, and they were: LT $= 0$ and HT $= 6.234375$. Based on these thresholds, we designed a single LUT for the sigmoid function with fractional bit $W_0 = 8$ and the input value $x$ ranging from LT to HT for the sampling interval given as $j = f = 8$. Using (13) and (1), the depth and the total size of the single LUT were $D_0 = 1597$ bits and $T_0 = 12\,776$ bits, respectively.

To design the t-LUT, recall (9) and Fig. 3. When $B = 4$ or $B = 8$, the compressibility is maximal, and subsequently, we use the exhaustive method (see Appendix A) to calculate the actual $E$. For the sigmoid function, $E = 2$ when $B = 4$ and $E = 3$ when $B = 8$ and since ceil($\log_2 E$) is less than $\log_2 B$ and the compressibility can be improved from 50% to approximately 62.45% with a reduction of 7979 bits when $B = 4$ and $E = 2$ or from 50% to approximately 62.48% with a reduction of 7982 bits when $B = 8$ and $E = 3$, according to (5). The difference between ($B = 4, E = 2$) and ($B = 8, E = 3$) is caused by the approximation from (5) to (9). Therefore, if there are different $B$'s, they make the compressibility maximal according to (9) and Fig. 3, while an error may exist because of the approximation from (5) to (9).

For a comparison, we chose $B = 8$ and $E = 3$ and, thus, grouped eight neighboring data together and stored the minimum value across the neighbors in the d-LUT. Thus, the width of d-LUT is equal to $W_0$ and the depth of d-LUT $D_d$ is 200, based on (2). Subsequently, we calculate a

TABLE II
LUT SIZE OF DIFFERENT FUNCTIONS

| Function | Size of Single LUT $W_0 \times D_0(bits)$ | Size of t-LUT $W_e \times D_e + W_d \times D_d(bits) = ceil(\log_2 E) \times D_0 + W_0 \times ceil(\frac{D_0}{B})(bits)$ | Compressibility | Range | Maximum Absolute Error | Mean Squared Error |
|---|---|---|---|---|---|---|
| sigmoid | $8 \times 1597 = 12776$ | $ceil(log_2 3) \times 1597 + 8 \times ceil(\frac{1597}{8}) = 4794$ | 62.48% | [0, 6.234375] | 0.001953 | 1.226473e-06 |
| hyperbolic tangent | $8 \times 888 = 7104$ | $ceil(log_2 4) \times 888 + 8 \times ceil(\frac{888}{4}) = 3552$ | 50.00% | [0, 3.46484375] | 0.001952 | 6.521651e-07 |
| ELU | $8 \times 1597 = 12776$ | $ceil(log_2 5) \times 1597 + 8 \times ceil(\frac{1597}{8}) = 6391$ | 49.98% | [-6.23828125, -0.00390625] | 0.001953 | 1.281475e-06 |
| softsign | $8 \times 2048 = 16384$ | $ceil(log_2 4) \times 2048 + 8 \times ceil(\frac{2048}{4}) = 8192$ | 50.00% | [0, 7.99609375] | 0.001951 | 1.271225e-06 |
| softplus | $12 \times 2048 = 24576$ | $ceil(log_2 5) \times 2048 + 12 \times ceil(\frac{2048}{8}) = 9216$ | 62.50% | [-4, 3.99609375] | 0.001952 | 1.263816e-06 |
| $\frac{x}{\sqrt{1+x^2}}$ | $8 \times 2048 = 16384$ | $ceil(log_2 4) \times 2048 + 8 \times ceil(\frac{2048}{4}) = 8192$ | 50.00% | [0, 7.99609375] | 0.001953 | 1.244180e-06 |
| $erf(\frac{\sqrt{\pi}}{2}x)$ | $8 \times 633 = 5064$ | $ceil(log_2 4) \times 633 + 8 \times ceil(\frac{633}{4}) = 2538$ | 49.88% | [0, 2.46875] | 0.001950 | 4.323394e-07 |
| $\frac{2}{\pi}\arctan(\frac{\pi}{2}x)$ | $8 \times 2048 = 16384$ | $ceil(log_2 4) \times 2048 + 8 \times ceil(\frac{2048}{4}) = 8192$ | 50.00% | [0, 7.99609375] | 0.001952 | 1.258111e-06 |
| GELU | $12 \times 2048 = 24576$ | $(ceil(log_2 5) + 1) \times 2048 + 12 \times ceil(\frac{2048}{8}) = 11264$ | 54.17% | [-4, 3.99609375] | 0.001952 | 1.185720e-06 |
| $exp(x)$ | $15 \times 1242 = 18630$ | failed | 00.00% | [0, 4.84765625] | 0.001953 | 1.240687e-06 |

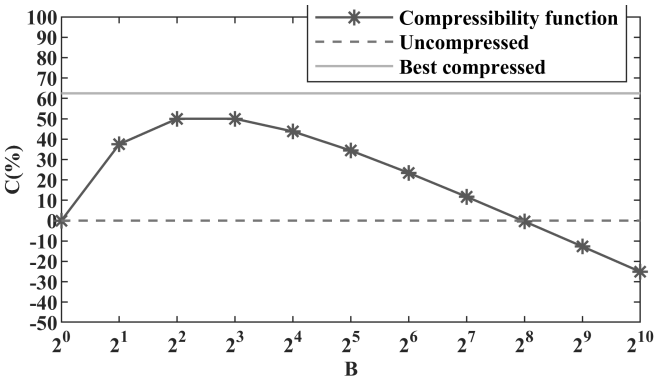



Fig. 5. Sigmoid-case compressibility function with the best 62.48% compressibility when $E = 3$ and $B = 8$.

difference between the original value and the stored minimum value and use ceil($\log_2 E$) = 2 bits to store the error in the e-LUT. Therefore, the width of e-LUT is 2 and the depth of e-LUT is 1597 according to (4) and (3). Fig. 5 shows the compressibility of sigmoid function. In Table II, more activation functions were approximated using the t-LUT structure and then compared with a single structure (GELU and exp($x$) are discussed in Section V).

## IV. HARDWARE IMPLEMENTATION AND UTILIZATION ANALYSIS

To establish the utilized hardware resources, we implemented the proposed LUT with t-LUT structure and compared it with the single LUT structure. The implementations were done on a Xilinx Spartan-7 XC7S50 FPGA device with 18k-bits Block RAM. Fig. 6 shows the structure of our t-LUT.

### A. Sketch of the Hardware Architecture

The proposed LUT with t-LUT architecture consists of four multiplexers, one comparator, three adders/subtractors units, two core LUTs (d-LUT and e-LUT), and several buffers for synchronization. Assuming that the LUTs are preloaded, the following steps are employed to retrieve the stored values. When queried with a 12-bit input [x(11:0)], for a value greater than 0 [x(11) = 0)] (refer to Fig. 6), the absolute value of the input [x(10:0)] is compared (using the comparator CMP) to verify the conditions mentioned in (10). Now, we analyze two possible cases.

TABLE III
BIT WIDTH OF T-LUT

| Function | Address Width of d-LUT | Address Width of e-LUT | Data Width of d-LUT | Data Width of e-LUT |
|---|---|---|---|---|
| sigmoid | 8 | 11 | 8 | 2 |
| tanh | 8 | 10 | 8 | 2 |
| softsign | 9 | 11 | 8 | 2 |

*Case 1:* The input is within HT, i.e., $0 \leq x \leq$ HT. Now, the comparator enables the LUTs (ena = 1 in Fig. 6) and the input is presented as an address to retrieve the preloaded LUT data from both d-LUT and e-LUT. The depth and width of the LUTs differ for different activation functions. For the sigmoid function, d-LUT and e-LUT require an 8-bit and an 11-bit addressing and the output value to have 8 and 2 bits, respectively. The address width and the data width for a few activation functions, such as tanh and softsign are presented in Table III. Once the data are retrieved from the LUTs, to satisfy the generalized bit-width requirements, the retrieved values are concatenated with zeros and summed and presented as 12-bit outputs.

*Case 2:* The input exceeds HT ($x >$ HT). The comparator disables d-LUT and e-LUT and employs the multiplexers (MUX2 and MUX3) to present a 12-bit constant value $C$ as the output. Also, when a negative input is presented (x(11) = 1), the input is 2′s complemented and the absolute value is presented to comparator through MUX1, which retrieves the 12-bit preloaded data based on the scenarios presented earlier (cases 1 and 2). Furthermore, to compensate for the symmetry, the 12-bit output is subtracted from the contact value $C$ and presented as the output through MUX4, as shown in Fig. 6.

### B. Utilization

Tables IV–VI present the utilization information for different activation functions, based on the proposed t-LUT method.

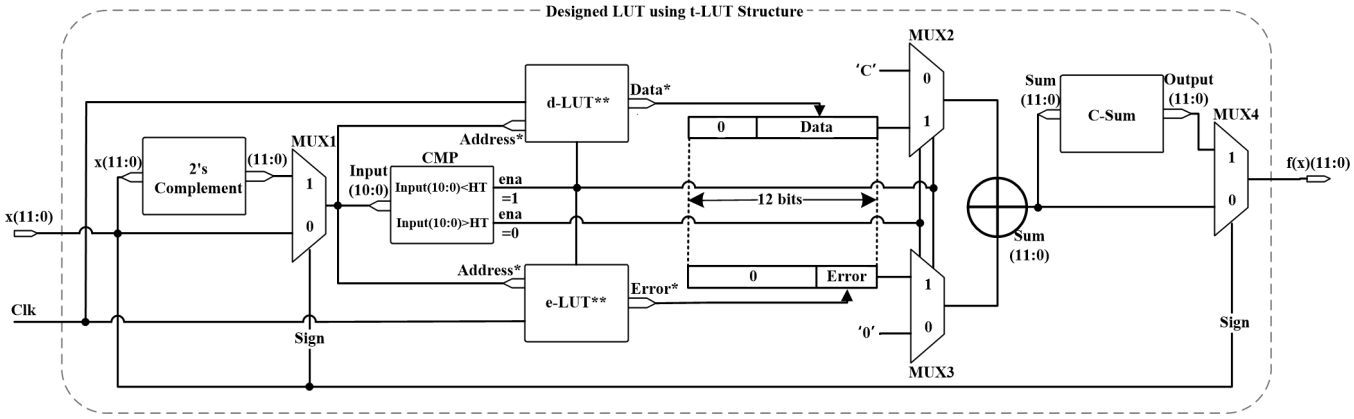



Fig. 6. Designed circuit for sigmoid, tanh, softsign, $(x/(1+x^2)^{1/2})$, erf$((\sqrt{\pi}/2)x)$, and $(2/\pi)\arctan((\pi/2)x)$ functions using t-LUT in FPGA. Note: * stands for the corresponding bit width of various functions. ** stands for the corresponding depth of d-LUT and e-LUT. Table III shows address width and the data width for sigmoid, tanh, and softsign.

TABLE IV
UTILIZATION SUMMARY 1 OF ACTIVATION FUNCTION

| Activation Function | sigmoid | | tanh | | softsign | |
|---|---|---|---|---|---|---|
| Architecture | t-LUT Used | Single LUT Used | t-LUT Used | Single LUT Used | t-LUT Used | Single LUT Used |
| Slice LUTs | 39 | 23 | 38 | 22 | 40 | 23 |
| Used as logic | 39 | 23 | 38 | 22 | 40 | 23 |
| Occupied Slices | 14 | 7 | 13 | 7 | 15 | 7 |
| Carry4 | 5 | 2 | 5 | 2 | 5 | 2 |
| 18k BRAMs | 3,194 | 12,776 | 1,776 | 3,552 | 4,096 | 8,192 |
| (bits) | +1,600 | | +1,776 | | +4096 | |

TABLE V
UTILIZATION SUMMARY 2 OF ACTIVATION FUNCTION

| Activation Function | $\frac{x}{\sqrt{1+x^2}}$ | | $erf(\frac{\sqrt{\pi}}{2}x)$ | | $\frac{2}{\pi}\arctan(\frac{\pi}{2}x)$ | |
|---|---|---|---|---|---|---|
| Architecture | t-LUT Used | Single LUT Used | t-LUT Used | Single LUT Used | t-LUT Used | Single LUT Used |
| Slice LUTs | 38 | 22 | 37 | 21 | 39 | 23 |
| Used as logic | 38 | 22 | 37 | 21 | 39 | 23 |
| Occupied Slices | 13 | 8 | 14 | 8 | 14 | 8 |
| Carry4 | 5 | 2 | 5 | 2 | 5 | 2 |
| 18k BRAMs | 4,096 | 16,384 | 1,266 | 5,064 | 4,096 | 16,384 |
| (bits) | +4,096 | | +1,272 | | +4,096 | |

Comparing with a single LUT structure to approximate a sigmoid function, the t-LUT consumes more logic gates for the design adders and the comparator, whereas in terms of Block RAM usage, the single LUT uses 12 776 bits, compared with the t-LUT structure requiring only 4794 bits, presenting a compressibility 62.48% with a reduction of 7982 bits.

## V. DISCUSSION

In this section, we will analyze the limitation and advantages of our proposed t-LUT and further present comparisons in terms of the hardware utilization with respect to other existing LUT design techniques.

### A. Limitation

Our previous results were based on the precondition that the difference between the two consecutive values in the original LUT should be either 0 or $\pm 1$ [see Fig. 7(a) and (c)]. When the precondition is not met [see Fig. 7(b) and (d)], the e-LUT exhibits an overflow, and hence, more bits are required to avoid the overflow, thus directly reducing the compressibility. The following cases illustrate the preconditions.

*Case 1:* The difference between the two consecutive values of $f(x)$ is neither 0 nor $\pm 1$. For example, while approximating the GELU function, we store the approximated function in an LUT for an input range $-4 \leq x \leq 3.99609375$, according to (11), before the function saturates. Also, a 12-bit fixed point structure with four signed integer bits and eight fractional bits was used to approximate the values of the function. In theory, the maximum compressibility $C = 62.50\%$ can be achieved with $B = 8$ and $W_e = 3$ (see Fig. 3), but as the difference between the consecutive values ranges from $-1$ to 2 [see Fig. 7(b)], it causes the e-LUT to overflow. To avoid the overflow, we have to increase the width of the e-LUT to store the whole error value, leading to a reduction in compressibility from 62.5% to roughly 54.17%. Table II shows that if the difference between the consecutive values is larger (as an example, the derivative of exponent function is also an exponential function where $(d(e^x)/dx) = e^x$), the width of the e-LUT will increase, which leads to a reduced compressibility or even uncompressed scenarios (refer to GELU and $\exp(x)$ in Table II).

TABLE VI
UTILIZATION SUMMARY 3 OF ACTIVATION FUNCTION (SEE APPENDIX C)

| Activation Function | ELU | | softplus | | GELU | |
|---|---|---|---|---|---|---|
| Architecture | t-LUT Used | Single LUT Used | t-LUT Used | Single LUT Used | t-LUT Used | Single LUT Used |
| Slice LUTs | 33 | 27 | 22 | 17 | 24 | 18 |
| Used as logic | 33 | 27 | 22 | 17 | 24 | 18 |
| Occupied Slices | 11 | 7 | 6 | 8 | 8 | 7 |
| Carry4 | 7 | 4 | 5 | 2 | 5 | 2 |
| 18k BRAMs | 4,791 | 12,776 | 6,144 | 24,576 | 8,192 | 24,576 |
| (bits) | +1,600 | | +3,072 | | +3,072 | |

TABLE VII

COMPLEXITY COMPARISON OF DIFFERENT IMPLEMENTATIONS OF HYPERBOLIC TANGENT WITH 0.04 MAXIMUM ERROR

| Architectures | Depth of LUT | Width of LUT | Size of LUT(bits) | Compresibility | BRAM Utilization | Maximum Error | Average Error | Range |
|---|---|---|---|---|---|---|---|---|
| LUT [11] | 512 | 9 | 4608 | 0.00% | 25.00% | 0.036500 | 0.004000 | [-8,8) |
| RALUT [11] | 61 | 9 | 549 | 88.09% | 2.98% | 0.035700 | 0.008900 | [-8,8) |
| RALUT+t-LUT | $9^{!}$ $65^{*}$ | $9^{!}$ $3^{*}$ | 276 | 94.01% | 1.50% | 0.015322 | 0.003203 | [-8,8) |
| t-LUT | $20^{!}$ $78^{*}$ | $5^{!}$ $2^{*}$ | 256 | 94.44% | 1.39% | 0.015322 | 0.003203 | [0,2.40625] |
| t-LUT+RALUT** | $8^{!}$ $32^{*}$ | $5^{!}$ $2^{*}$ | 104 | 97.74% | 0.56% | 0.015322 | 0.003203 | [0,2.40625] |
| Hybrid Architecture [5] | 16 | 5 | 80 | 98.26% | 0.43% | 0.037800 | – | (0.5,2) |
| Hybrid Architecture [5]+t-LUT | $4^{!}$ $16^{*}$ | $5^{!}$ $3^{*}$ | 68 | 98.52% | 0.37% | 0.037800 | – | (0.5,2) |

! stands for d-LUT. $*$ stands for e-LUT. $**$ RALUT+t-LUT (line3) and t-LUT+RALUT (line5) are different in range. LUT [11] is the baseline.

TABLE VIII

COMPLEXITY COMPARISON OF DIFFERENT IMPLEMENTATIONS OF HYPERBOLIC TANGENT WITH 0.02 MAXIMUM ERROR

| Architectures | Depth of LUT | Width of LUT | Size of LUT(bits) | Compresibility | BRAM Utilization | Maximum Error | Average Error | Range |
|---|---|---|---|---|---|---|---|---|
| LUT [11] | 1024 | 10 | 10240 | 0.00% | 55.56% | 0.018000 | 0.002000 | [-8,8) |
| RALUT [11] | 127 | 10 | 1270 | 87.60% | 6.89% | 0.017800 | 0.005700 | [-8,8) |
| RALUT+t-LUT | $17^{!}$ $129^{*}$ | $10^{!}$ $3^{*}$ | 557 | 94.56% | 3.02% | 0.007790 | 0.001794 | [-8,8) |
| t-LUT | $45^{!}$ $178^{*}$ | $6^{!}$ $2^{*}$ | 626 | 93.89% | 3.40% | 0.007790 | 0.001794 | [0,2.765625] |
| t-LUT+RALUT** | $16^{!}$ $64^{*}$ | $6^{!}$ $2^{*}$ | 224 | 97.81% | 1.22% | 0.007790 | 0.001794 | [0,2.765625] |
| Hybrid Architecture [12] | 15 | 8 | 120 | 98.83% | 0.65% | 0.020000 | – | [0.390625,2.90625] |
| Hybrid Architecture [12]+t-LUT | $4^{!}$ $15^{*}$ | $8^{!}$ $5^{*}$ | 107 | 98.96% | 0.58% | 0.020000 | – | [0.390625,2.90625] |

! stands for d-LUT. $*$ stands for e-LUT. $**$ RALUT+t-LUT (line3) and t-LUT+RALUT (line5) are different in range. LUT [11] is the baseline.

TABLE IX

COMPLEXITY COMPARISON OF DIFFERENT IMPLEMENTATIONS OF SIGMOID WITH 0.02 MAXIMUM ERROR

| Architectures | Depth of LUT | Width of LUT | Size of LUT(bits) | Compresibility | BRAM Utilization | Maximum Error | Average Error | Range |
|---|---|---|---|---|---|---|---|---|
| LUT | 512 | 9 | 4608 | 0.00% | 25.00% | 0.015620 | 0.006028 | [-8,8) |
| RALUT | 33 | 9 | 297 | 93.55% | 1.61% | 0.015620 | 0.006028 | [-8,8) |
| RALUT+t-LUT | $5^{!}$ $33^{*}$ | $9^{!}$ $3^{*}$ | 144 | 96.86% | 0.78% | 0.015620 | 0.006028 | [-8,8) |
| t-LUT | $34^{!}$ $133^{*}$ | $5^{!}$ $1^{*}$ | 303 | 93.42% | 1.64% | 0.015620 | 0.006028 | [0,4.125] |
| t-LUT+RALUT** | $4^{!}$ $16^{*}$ | $5^{!}$ $2^{*}$ | 52 | 98.87% | 0.28% | 0.015620 | 0.006028 | [0,4.125] |

! stands for d-LUT. $*$ stands for e-LUT. $**$ RALUT+t-LUT (line3) and t-LUT+RALUT (line5) are different in range. LUT (line1) is the baseline.

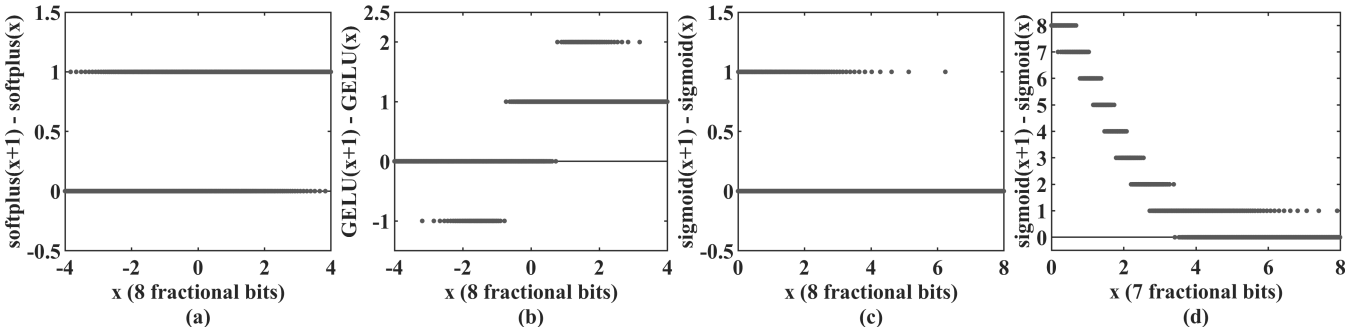



Fig. 7. (a) Difference of LUT approx. for eight-fractional-bit softplus is either 0 or 1. (b) Difference of LUT approx. for eight-fractional-bit GELU ranges from $-1$ to 2. (c) Difference of LUT approx. for eight-fractional-bit sigmoid is either 0 or 1. (d) Difference of LUT approx. for 12-fractional-bit sigmoid ranges from 0 to 8.

*Case 2:* A large sampling interval [referring to (12)] presents a difference between the consecutive values of $f(x)$ as being neither 0 nor $\pm 1$. As shown in Table I (with the analysis based on an 18k-bit memory), as $f$ increases from 4 to 8, $j$ becomes equal to $f$, and the compressibility increases from 24.09% to 62.48% since the consecutive values are either 0 or $\pm 1$ [see Fig. 7(c)]. The same is valid when $f = 9$ and $j = 8$. However, when $f$ increases from 10 to 12, keeping $j$ fixed to 7, the differences between the consecutive values of $f(x)$

TABLE X
COMPRESSIBILITY OF DIFFERENT WIDTHS AND BAND SIZES

| Width \ Band Size | 2 | 4 | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 |
|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 0.00% | <0 | <0 | <0 | <0 | <0 | <0 | <0 | <0 | <0 |
| 3 | 16.67% | 8.33% | <0 | <0 | <0 | <0 | <0 | <0 | <0 | <0 |
| 4 | 25.00% | 25.00% | 12.50% | <0 | <0 | <0 | <0 | <0 | <0 | <0 |
| 5 | 30.00% | 35.00% | 27.50% | 13.75% | <0 | <0 | <0 | <0 | <0 | <0 |
| 6 | 33.33% | 41.67% | 37.50% | 27.08% | 13.54% | <0 | <0 | <0 | <0 | <0 |
| 7 | 35.71% | 46.43% | 44.64% | 36.61% | 25.45% | 12.72% | <0 | <0 | <0 | <0 |
| 8 | 37.50% | 50.00% | 50.00% | 43.75% | 34.38% | 23.44% | 11.72% | <0 | <0 | <0 |
| 9 | 38.89% | 52.78% | 54.17% | 49.31% | 41.32% | 31.77% | 21.44% | 10.72% | <0 | <0 |
| 10 | 40.00% | 55.00% | 57.50% | 53.75% | 46.88% | 38.44% | 29.22% | 19.61% | 9.80% | <0 |
| 11 | 40.91% | 56.82% | 60.23% | 57.39% | 51.42% | 43.89% | 35.58% | 26.88% | 17.99% | 8.99% |
| 12 | 41.67% | 58.33% | 62.50% | 60.42% | 55.21% | 48.44% | 40.89% | 32.94% | 24.80% | 16.57% |
| 13 | 42.31% | 59.62% | 64.42% | 62.98% | 58.41% | 52.28% | 45.37% | 38.07% | 30.57% | 22.98% |
| 14 | 42.86% | 60.71% | 66.07% | 65.18% | 61.16% | 55.58% | 49.22% | 42.47% | 35.52% | 28.47% |
| 15 | 43.33% | 61.67% | 67.50% | 67.08% | 63.54% | 58.44% | 52.55% | 46.28% | 39.80% | 33.24% |
| 16 | 43.75% | 62.50% | 68.75% | 68.75% | 65.63% | 60.94% | 55.47% | 49.61% | 43.55% | 37.40% |
| 32 | 46.88% | 68.75% | 78.13% | 81.25% | 81.25% | 79.69% | 77.34% | 74.61% | 71.68% | 68.65% |
| 64 | 48.44% | 71.88% | 82.81% | 87.50% | 89.06% | 89.06% | 88.28% | 87.11% | 85.74% | 84.28% |

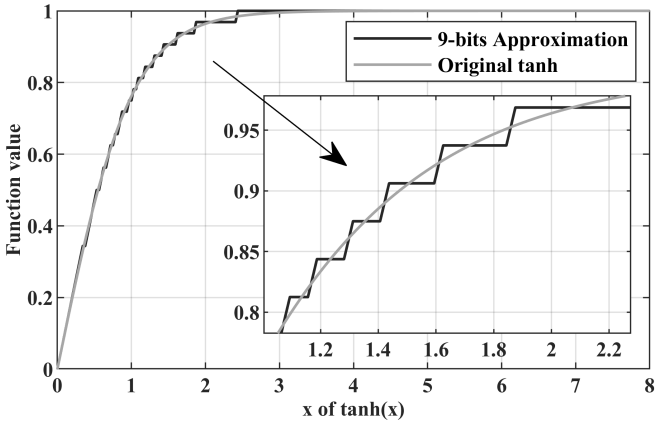



Fig. 8. Comparison of the original tanh function and different approximations.

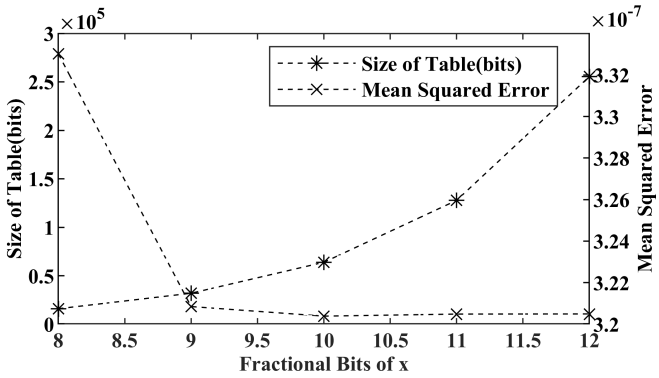



Fig. 9. Experimental analysis for $f = 9$ [i.e., fractional bits of $F(x)$] and $j = 8, 9, 10, 11, 12$ (i.e., fractional bits of $x$), which clearly shows that when $j > f$, it does not reduce the MSE but exponentially increases the necessary hardware, resulting in a huge memory utilization.

do not satisfy the precondition [the difference ranges from 0 to 8, as shown in Fig. 7(d)], thus resulting in lower compressibility. Furthermore, even when $f > 12$, the differences between the consecutive values of $f(x)$ do not satisfy the precondition and the compressibility decreases.

### B. Advantages

1) We used t-LUT to approximate different activation functions, as shown in Table II. In the respective literature, there are eight popular activation functions [from sigmoid to $(2/\pi)\arctan((\pi/2)x)$] that meet our precondition and reach the theoretic compressibility. Among these results, the softsign reaches the maximum compressibility of 62.5% when compared with the use of a single LUT.
2) Since the compressibility is independent of the LUT depth [as shown in (8)], an increase in the number of entries in the t-LUT will proportionally increase the size of the LUTs (d-LUT and e-LUT) with no additional logic added to the existing circuitry. In contrast, the hardware resources in the implementations in [9], [10], and [12], such as logic gates and comparators, increase proportionally with the number of additional entries, on top of the increase in the required memory resources.
3) The proposed design employs an e-LUT to avoid any error and thus compresses the information without any loss, thereby approximating the activation function with high-precision and less memory resources.

### C. Comparison

In order to provide realistic comparisons, we chose to approximate a hyperbolic tangent function, tanh, with a 3.91% relative error and a 3.070168e-05 MSE, as shown in Fig. 8. Furthermore, we also set the LUT architecture designed in [11] as our benchmark and calculate the compressibility and the resources utilization in terms of 18k bits of Block RAM for different architectures. The results are presented in Tables VII and VIII for the LUT depth size of 512 and 1024, respectively. RALUT in [11] utilized 549 bits (2.98%) and 1270 bits (6.89%) of the Block RAM and presented the

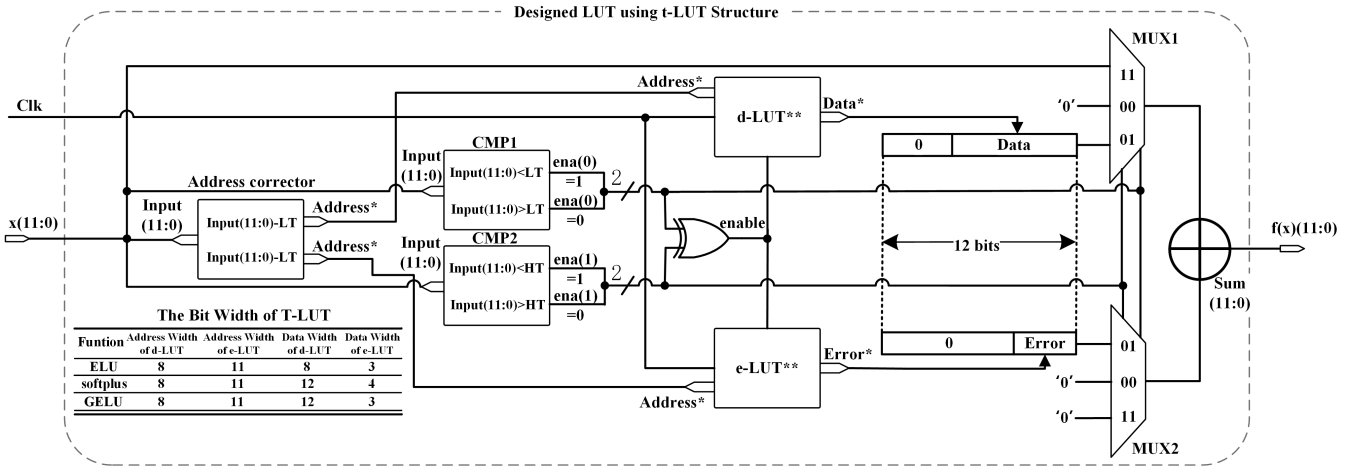



Fig. 10. Designed circuit for ELU, softplus, and GELU activation functions using the t-LUT in an FPGA. Note: * stands for the corresponding bit width of various functions. ** stands for the corresponding depth of d-LUT and e-LUT. Table in Fig. 10 shows the address width and the data width for ELU, softplus, and GELU.

TABLE XI
ERROR ANALYSIS OF DIFFERENCE SAMPLING PRECISIONS

| Fractional Bits | | Sampling | Size of | Maximal | Average | Mean |
|---|---|---|---|---|---|---|
| F(x) | x | Interval(E-04) | Table (bits) | Error(E-04) | Error(E-04) | Squared(E-07) |
| 9 | 8 | 39.0625000 | 15975 | 9.765613 | 5.064665 | 3.330143 |
| 9 | 9 | 19.5312500 | 31941 | 9.765613 | 4.925943 | 3.208491 |
| 9 | 10 | 9.76562500 | 63873 | 9.765613 | 4.924733 | 3.203886 |
| 9 | 11 | 4.88281250 | 127746 | 9.765613 | 4.925514 | 3.204872 |
| 9 | 12 | 2.44140625 | 255492 | 9.765613 | 4.925671 | 3.204941 |

compressibility of 88.09% and 87.60%, for the LUT sizes of 512 and 1024, respectively. In comparison, for the same sizes of the LUT, our t-LUT presented a higher compressibility of 94.44% and 93.89%. The abovementioned results clearly indicate that the proposed t-LUT design is effective in reducing hardware when compared with similar LUT implementations. Furthermore, RALUT in [11] stores only unique values of the activation function in each memory location and is based on a careful analysis. It was noticed that the difference between the two consecutive values in [11] differs by 1, thus satisfying our precondition. Clearly, using our proposed t-LUT instead of the LUT used in [11] improved the respective compressibilities for about 6%. For the implementation presented in [5], for a depth size of 16, the compressibility of 98.26% was obtained when compared with LUT [11]. Another hybrid method presented in [12] used custom logic circuits along with an LUT size of 120 bits (0.65%) and reported a compressibility of 98.83% when compared with the LUT architecture defined in [11]. Replacing the LUTs in [5] and [12] by the proposed t-LUT improved the respective compressibility to 98.52% and 98.96%, respectively. Therefore, LUT implementations satisfying our preconditions can be directly replaced by our t-LUT, attaining even higher compressibility. Furthermore, we have also approximated the sigmoid function using our t-LUT and compared it to the conventional LUT architecture and RALUT architecture in Table IX. Here, the conventional LUT that utilized 4608 bits (25.00%) was used as a benchmark for the comparison. The RALUT architecture utilized 297 bits (1.61%) and presented a compressibility of 93.55%. When replaced with t-LUT, it used 303 bits (1.64%) and reported a compressibility of 93.42%. Also, the combined architecture of t-LUT and RALUT cost 144 bits (0.78%) and reached an improved compressibility of 96.86%.

**Algorithm 1** Exhaustive Method Finding E in Original LUT

**Input:** LUT array $lut$, Band size $B$
**Output:** the maximal different number in one band $dnum\_max$
**for** $i = every\ indexes\ of\ lut$ **do**
  $k = i + B$
  **if** $i \leq length(lut) - B$ **then**
    $temp = lut(i:k)$
    **if** $mod(i, B) == 0$ **then**
      $dnum(end + 1) = length(unique(temp))$
    **end if**
  **else**
    $temp = the\ rest\ part\ of\ lut$
    $dnum(end + 1) = length(unique(temp))$
    $break$
  **end if**
**end for**
$dnum\_max = max(dnum)$

Overall, when the performance of the conventional LUT is compared against the proposed t-LUT in terms of the BRAM utilization column data in Tables VII–IX, the t-LUT outperforms the LUT by the factor greater than 16.

## VI. CONCLUSION

LUT implementations on FPGAs are commonly used in NNs to approximate the activation functions. In this article, we proposed a t-LUT architecture, which employs two parallel LUTs to approximate different activation functions. In our deployments, implementation of the t-LUT requires less resources and has a better performance compared with

previously implemented LUT structures. The proposed design can be efficiently used in NN implementations on an FPGA.

Furthermore, any existing LUT architectures, satisfying the described preconditions, can be replaced by the proposed t-LUT, consequently achieving higher compressibility rates. We believe that the method can be applied to other function approximations, e.g., square-root, logarithm, and trigonometric functions in digital signal processors (DSPs) or GPUs.

## APPENDIX A
## PSEUDO CODE

See Algorithm 1.

## APPENDIX B
## SIZE OF TABLE AND MSE

See Fig. 9 and Table XI.

## APPENDIX C
## CIRCUIT FIGURE

See Fig. 10.

## REFERENCES


[1] C.-W. Lin and J.-S. Wang, "A digital circuit design of hyperbolic tangent sigmoid function for neural networks," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2008, pp. 856–859.

[2] K. Basterretxea, J. M. Tarela, and I. del Campo, "Approximation of sigmoid function and the derivative for hardware implementation of artificial neurons," *IEE Proc.-Circuits, Devices Syst.*, vol. 151, no. 1, pp. 18–24, Feb. 2004.

[3] F. Piazza, A. Uncini, and M. Zenobi, "Neural networks with digital LUT activation functions," in *Proc. Int. Conf. Neural Netw.*, Oct. 1993, pp. 1401–1404.

[4] J. G. Delgado-Frias, M. Zhang, and S. Vassiliadis, "Elementary function generators for neural-network emulators," *IEEE Trans. Neural Netw.*, vol. 11, no. 6, pp. 1438–1449, Nov. 2000.

[5] B. Zamanlooy and M. Mirhassani, "Efficient VLSI implementation of neural networks with hyperbolic tangent activation function," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 1, pp. 39–48, Jan. 2014.

[6] A. H. Namin, K. Leboeuf, R. Muscedere, H. Wu, and M. Ahmadi, "Efficient hardware implementation of the hyperbolic tangent sigmoid function," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2009, pp. 2117–2120.

[7] Z. Hajduk, "High accuracy FPGA activation function implementation for neural networks," *Neurocomputing*, vol. 247, pp. 59–61, Jul. 2017.

[8] F. Ortega-Zamorano, J. M. Jerez, G. Juárez, J. O. Pérez, and L. Franco, "High precision FPGA implementation of neural network activation functions," in *Proc. IEEE Symp. Intell. Embedded Syst. (IES)*, Dec. 2014, pp. 55–60.

[9] R. Muscedere, V. Dimitrov, G. A. Jullien, and W. C. Miller, "Efficient techniques for binary-to-multidigit multidimensional logarithmic number system conversion using range-addressable look-up tables," *IEEE Trans. Comput.*, vol. 54, no. 3, pp. 257–271, Mar. 2005.

[10] R. Muscedere and K. Leboeuf, "A dynamic address decode circuit for implementing range addressable look-up tables," in *Proc. IEEE Int. Symp. Circuits Syst.*, Seattle, WA, USA, May 2008, pp. 18–21.

[11] K. Leboeuf, A. H. Namin, R. Muscedere, H. Wu, and M. Ahmadi, "High speed VLSI implementation of the hyperbolic tangent sigmoid function," in *Proc. 3rd Int. Conf. Converg. Hybrid Inf. Technol.*, Nov. 2008, pp. 1070–1073.

[12] P. Kumar Meher, "An optimized lookup-table for the evaluation of sigmoid function for artificial neural networks," in *Proc. 18th IEEE/IFIP Int. Conf. VLSI System-on-Chip*, Madrid, Spain, Sep. 2010, pp. 27–29.

[13] E. Reinhard, E. Garces, and J. Stauder, "Repeated look-up tables," *IEEE Trans. Image Process.*, vol. 29, pp. 2370–2379, 2020.

[14] T. Bonny and J. Henkel, "Efficient code density through look-up table compression," in *Proc. Des., Automat. Test Eur. Conf. Exhib.*, Apr. 2007, pp. 1–6.

[15] S.-F. Hsiao, P.-H. Wu, C.-S. Wen, and P. K. Meher, "Table size reduction methods for faithfully rounded lookup-table-based multiplierless function evaluation," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 62, no. 5, pp. 466–470, May 2015.

[16] D. Das Sarma and D. W. Matula, "Faithful bipartite ROM reciprocal tables," in *Proc. 12th Symp. Comput. Arithmetic*, 1995, pp. 17–28.

[17] M. J. Schulte and J. E. Stine, "Approximating elementary functions with symmetric bipartite tables," *IEEE Trans. Comput.*, vol. 48, no. 8, pp. 842–847, Aug. 1999.

[18] N. Koc-Sahan, J. Schlessman, and M. J. Schulte, "Symmetric table addition methods for neural network approximations," *Proc. SPIE*, vol. 4474, pp. 126–133, Nov. 2001.

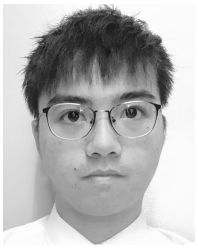
[19] F. de Dinechin and A. Tisserand, "Multipartite table methods," *IEEE Trans. Comput.*, vol. 54, no. 3, pp. 319–330, Mar. 2005.


**Yusheng Xie** is currently working toward the B.S. degree in electronic information engineering at Shantou University, Guangdong, China.

His research interests include neural network implementation on field-programmable gate arrays (FPGAs), image processing, and computer vision.

**Alex Noel Joseph Raj** received the B.E. degree in electrical engineering from Madras University, Chennai, India, in 2001, the M.E. degree in applied electronics from Anna University, Chennai, in 2005, and the Ph.D. degree in engineering from the University of Warwick, Coventry, U.K., in 2009.

From October 2009 to September 2011, he was a Design Engineer with Valeport LTD Totnes, Totnes, U.K. From March 2013 to March 2017, he was with the Department of Embedded Technology, School of Electronics Engineering, Vellore Institute of Technology, Vellore, India, as a Professor. Since January 2017, he has been with the Department of Electronic Engineering, College of Engineering, Shantou University, Shantou, China. His research interests include deep learning, signal and image processing, and field-programmable gate array (FPGA) implementations.

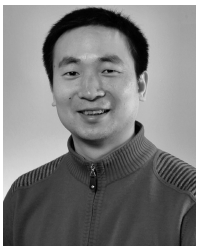


**Zhendong Hu** is currently working toward the B.S. degree in electronic information engineering at the Department of Electronic Engineering, Shantou University, Guangdong, China.

His research field involves the implementation of peripheral control and neural network on field-programmable gate arrays (FPGAs).

**Shaohaohan Huang** is currently working toward the B.S. degree at Shantou University, Guangdong, China, who is working for his double major in electronic engineering and business administration.

His research interests include machine learning, convolution neural networks, and field-programmable gate arrays (FPGAs).



**Zhun Fan** (Senior Member, IEEE) received the B.S. and M.S. degrees in control engineering from the Huazhong University of Science and Technology, Wuhan, China, in 1995 and 2000, respectively, and the Ph.D. degree in electrical and computer engineering from Michigan State University, Lansing, MI, USA, in 2004.

He is currently a Full Professor with Shantou University (STU), Shantou, China. He also serves as the Head of the Department of Electrical Engineering and the Director of the Guangdong Provincial Key Laboratory of Digital Signal and Image Processing. Before joining STU, he was an Associate Professor with the Technical University of Denmark (DTU), Kongens Lyngby, Denmark, from 2007 to 2011, where he was first with the Department of Mechanical Engineering and then with the Department of Management Engineering, and as an Assistant Professor with the Department of Mechanical Engineering from 2004 to 2007. He has been a Principle Investigator of a number of projects from the Danish Research Agency of Science Technology and Innovation and the National Natural Science Foundation of China. His research is also supported by the National Science Foundation. His major research interests include intelligent control and robotic systems, robot vision and cognition, MEMS, computational intelligence, design automation, optimization of mechatronic systems, machine learning, and image processing.

**Miroslav Joler** (Senior Member, IEEE) received the B.S. degree in electrical engineering from the University of Zagreb, Zagreb, Croatia, in 1996, and the M.S. and Ph.D. degrees in electrical engineering from The University of New Mexico, Albuquerque, NM, USA, in 2001 and 2006, respectively.

In 2006, he was a Postdoctoral Research Associate at Portland State University, Portland, OR, USA. In 2007, 2013, and 2018, he was elevated to Assistant Professor, Associate Professor, and Full Professor at the Faculty of Engineering, University of Rijeka, Rijeka, Croatia, respectively, where he also worked as an Adjunct Assistant Professor with the Faculty of Maritime Studies in 2008. Since 2008, he has been the Laboratory Director, the Head of the Communications Systems Group, the Graduation Committee Chair, and the Department Chair. He has been a Researcher in multiple scientific projects in the USA and Croatia, while his industry experience includes working as an RF Engineer from 1996 to 1999. He has coauthored articles in distinguished scientific journals and conferences. His research interests include antennas, smart clothing, wearable devices, self-recoverable systems, wireless power transfer, and biomedical applications of electromagnetics.

Dr. Joler served as a reviewer, a technical program committee member, and the session (co-)chair and held invited conference talks. He has also served as a Research Proposal and Annual Report Evaluator, an Editorial (Advisory) Board Member, and an Associate Editor. In 2001, he received the University of New Mexico Graduate Office's Research, Proposal, and Travel Award. In 2017, he was admitted to the Associate Level of the Croatian Academy of Engineering.