An Efficient LSI Implementation of the Summation of Products in Convolution Operation for Binarized Neural Networks

Mitsuru Takahashi Kazuhito Ito

Graduate School of Science and Engineering
Saitama University
Saitama 338-8570, Japan

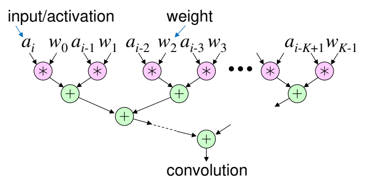
Abstract— Various applications of machine learning with convolutional neural networks (CNN) are emerging. A binarized NN (BNN) is a CNN where the number of bits of input, activation, and weight for convolution is one. Hence the inference operation in BNNs is simple and it is suitable for LSI implementation. In this paper, an efficient LSI implementation of the summation of the products in BNNs is proposed. The required number of transistors is reduced by 32% for the convolution kernel of the size $3 \times 3 \times 64$.

I. Introduction

The application of machine learning is expanding. The process of machine learning consists of learning and inference. Learning computes weights from a large amount of teacher data, and inference obtains results for input data using the weights obtained in learning.

Although the inference requires a large amount but relatively simple computations, these computations can be executed in parallel by dedicated hardware. By implementing inference in large scale integrated circuits (LSI), it will be possible to execute the inference in a small size, high speed, and low power consumption. That will enable to use machine learning results in mobile and IoT devices, and it is expected to further expand the application of machine learning. In this research, we examine the miniaturization of LSI that executes inference in machine learning.

Machine learning recently attracting attention uses a convolutional neural network (CNN) [1]. There is parallelism in the convolution operation between input or activation and weight, and it is possible to speed up the convolution by parallel processing. The number of bits of input, activation, and weight can be reduced to 1 bit to simplify the calculation of CNN, and it it called binarized CNN (BNN) [2]. When the number of bits is reduced to 1 bit, the multiplication can be realized by an exclusive NOR (XNOR) operation. Based on this advantage, LSI implementations of BNN has been proposed [3, 4, 5]. As a result of reducing the accuracy of input, activation, and weights to 1 bit in BNN, it is necessary to increase the number of weights (the order of the convolution kernel) to maintain the accuracy of inference, so the number of products to be summed up becomes large. In this research, we examine the method to reduce the number of transistors of the circuit for the summation of the products of input/activation and weight.



(a)

arithmetic			logic		
a	w	p	a	w	p
-1	-1	$+1$	0	0	1
-1	$+1$	-1	0	1	0
$+1$	-1	-1	1	0	0
$+1$	$+1$	$+1$	1	1	1

(b)

Fig. 1. Convolution and its implementation in BNNs. (a) convolution. (b) a 1-bit product.

XNOR circuits [6] and full adder (FA) circuits [7, 8, 9] consisting of a smaller number of transistors than a CMOS circuit have been proposed. Since these are not CMOS circuits, the output may take an intermediate voltage that does not reach the power supply voltage or the ground depending on the combination of input signal values. Since FAs are connected in multiple stages to sum up a large number of input/activation-weight products, the intermediate voltage output from an FA may be given to an FA in the next stage, and the intermediate voltages may cause misrecognition of logic values at the FA. In this research, we propose an efficient composition of FAs to obtain the summation of input/activation-weight products and output the result in a binary number. In addition, we propose FA circuits that requires the reduced number of transistors while the summation of the products are correctly calculated considering the intermediate voltages.

The remainder of the paper is organized as follows. The BNN and its necessary components are reviewed in Sect. 2. The proposed method is presented in Sect. 3. Experimental results are presented in Sect. 4 and Sect. 5 concludes the work.

II. Hardware Implementation of BNNs

A. *Convolution operation in BNNs*

The convolution operation of kernel size K is illustrated in Fig. 1(a). K inputs and K weights in the first layer or K activations and K weights in the second and later layers are multiplied one by one to get K products and these products are summed up. In BNN, the input/activation and weight take only two values, '$+1$' and '-1', and the combination of input/activation a and weight w values and their arithmetic product p are shown in the left half of Fig. 1(b). When the logic values 1 and 0 are assigned to '$+1$' and '-1', respectively, the truth table of the multiplication is as shown in the right half of

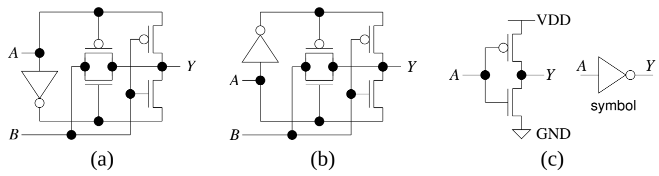


Fig. 2. 6 transistor XOR and XNOR gates. (a) XOR. (b) XNOR. (c) CMOS inverter.

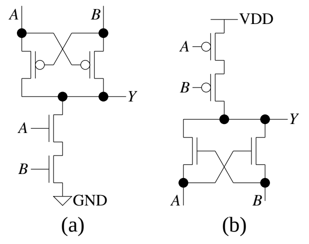


Fig. 3. 4 transistor XOR and XNOR gates. (a) XOR. (b) XNOR.

Fig. 1(b). Hence the 1-bit multiplication in BNNs can be performed by the inversion of the exclusive OR of *a* and *w*. That is, the 1-bit product can be obtained by an XNOR operation.

B. XOR and XNOR gates

In addition to the XNOR operation required for the above-mentioned input/activation and weight product, the XOR operation is required for the product summation. Various circuit configurations with different numbers of MOS transistors are known for XOR and XNOR gates [6]. Here the circuits with 6 and 4 transistors are briefly introduced.

The logical values 1 and 0 of the binary logic are represented respectively by a high voltage and a low voltage. It is said to be *strong 1* when the high voltage is equal to the positive power supply voltage (VDD) and *strong 0* when the low voltage is the ground (0 [V], GND). Unless otherwise specified, '1' represents a strong 1 and '0' represents a strong 0. Fig. 2(a) and Fig. 2(b) show respectively an XOR gate and an XNOR gate with 6 transistors. *A*, *B* are the inputs and *Y* is the output. The CMOS inverter shown in Fig. 2(c) is used in these gates. These 6T XOR / XNOR gates output a strong 1 or a strong 0 for every combination of the logic values 0 and 1 of the inputs *A* and *B*. The truth tables of the gates are shown in Table I.

Fig. 3 shows an XOR gate and an XNOR gate with 4 transistors. The 4T XOR gate in Fig. 3(a) has a strong 0 or strong 1 output when the input *A* is 1 and/or *B* is 1. When both inputs *A* and *B* are 0, the two PMOS transistors are both on and the charge at the output *Y* is discharged through the PMOS transistors towards the low voltage at the inputs *A* and *B*. When *Y* initially has a logical value of 1, the voltage of *Y* reduces as the discharge proceeds, and the PMOS transistor is cut off before *Y* reaches GND. The voltage is recognized as a logic value 0, and it is called *weak 0* and is expressed as 'w0'. Similarly, the 4T XNOR gate in Fig. 3(b) outputs 0 or 1 when the input *A* is 0 and/or *B* is 0. When both inputs *A* and *B* are 1, the voltage of *Y* may be lower than VDD although it is recognized as a logical value 1. This is called *weak 1* and is expressed as 'w1'. The truth tables of 4T XOR and XNOR gates are shown in Table I.

TABLE I
TRUTH TABLE OF XOR AND XNOR GATES

input		output			
		6 Tr		4 Tr	
A	*B*	XOR	XNOR	XOR	XNOR
0	0	0	1	w0	1
0	1	1	0	1	0
1	0	1	0	1	0
1	1	0	1	0	w1

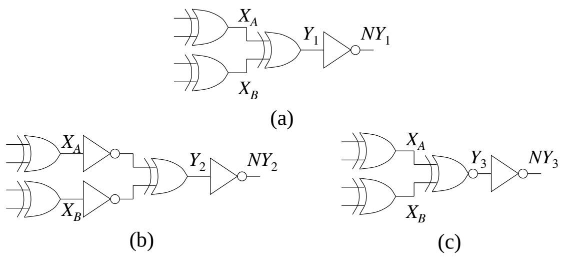


Fig. 4. The series connection of gates. (a) XOR gates followed by an XOR gate. (b) Restoring inverters are inserted. (c) XOR gates followed by an XNOR gate.

C. Consideration on weak 0 and weak 1

When logic circuits are connected in multiple stages as shown in Fig. 4, the weak 0 or weak 1 output by the previous gate may be given to the input of the next gate. Fig. 4(a) shows a circuit where XOR gates are connected in series. Here, the 4T XOR gate shown in Fig. 3(a) is used. In Fig. 4(a), a CMOS inverter is connected to the final output, which confirms whether the output Y_1 can be correctly recognized as logic 0 or logic 1. That is, if the output NY_1 of the inverter is logic 0, Y_1 is at logic 1, and if NY_1 is logic 1, Y_1 is at logic 0. The circuit simulation result is shown in Fig. 5. The target technology is 65 nm CMOS SOTB. The channel length $L = 60$ nm for both NMOS and PMOS transistors, and the channel width $W_n = 140$ nm and $W_p = 200$ nm for NMOS and PMOS transistors, respectively. The supply voltage VDD is 0.7 V, and the body bias is 0.3 V for NMOS and 0.4 V for PMOS transistors. The circuit simulation was performed using HSPICE. In Fig. 5, X_A is w0 at times 60 ns to 65 ns and 80 ns to 85 ns. X_B is w0 from time 70 ns to 90 ns. Immediately before time 85 ns, both X_A and X_B are w0, and from the input to output function of XOR, Y_1 is logic 0, and its inversion NY_1 must be logic 1. However, in the simulation result, NY_1 is logic 0 at that time. This indicates that the circuit is malfunctioning.

The cause of this malfunction is that a w0 in the output of the previous stage is further weakened in the next stage. The CMOS gate outputs a strong 1 or a strong 0 even if a w0 or a w1 is given as an input. This is called *restoration* of signal values. The circuit that restores the weak 0 outputs of X_A and X_B by CMOS inverters is shown in Fig. 4(b). From the circuit simulation results in Fig. 5, it can be seen that NY_2 and Y_2 are correctly obtained for all combinations of X_A and X_B.

A combination of gates that output w1s and w0s may save some intermediate restore inverters. Fig. 4(c) shows a circuit in which the second-stage XOR gate in Fig. 4(a) is replaced with the 4T XNOR. This XNOR gate outputs 1, 0, or w1 when the input values are 1, 0, or w0. That is, the input value of logic 0

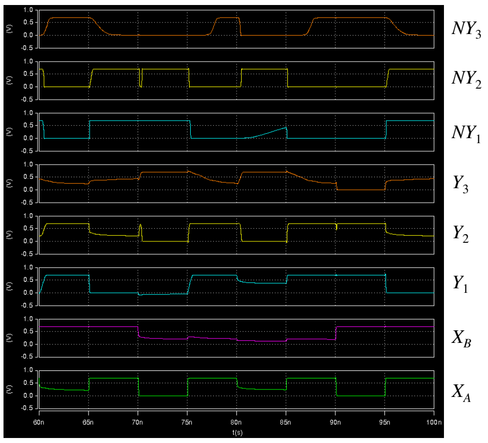


Fig. 5. The waveforms of the circuits shown in Fig. 4.

is not weakened. Therefore, when a w0 of the XOR gate output is given as an input, it is not further weakened. From the circuit simulation result of Fig. 5, it can be confirmed that Y_3 shows the correct logic values for all the input combinations of X_A and X_B, although the logic of the output Y_3 is inverted from Y_2 because the XOR gate is replaced with the XNOR gate.

The conclusion here is that if we connect a gate that outputs only w1 (w0) in addition to 0 and 1 to the next stage of the gate that outputs only w0 (w1) in addition to 0 and 1, it is not necessary to restore the intermediate signal values. By appropriately selecting a combination of w0 or w1 outputs in the previous stage and w1 or w0 outputs in the next stage, the restore inverter can be omitted and the number of transistors can be reduced accordingly.

D. *Conventional full adders*

Full adders (FAs) are used for bit addition of the products of input/activation and weight in BNN. Full adder circuit configurations with different numbers of transistors have been proposed, and some of them are reviewed here.

Fig. 6(a) shows the gate level circuit of an FA. *A*, *B*, *C* are inputs, Y_S is a sum output, and Y_C is a carry output. Two 6T XOR gates in Fig. 2(a) with $6 \times 2 = 12$ transistors, a CMOS AOI gate with 8 transistors, and a CMOS inverter with 2 transistors are employed and a total of 22 transistors are used.

Fig.6(b) shows a full adder that uses 14 transistors [7]. This circuit uses a 4T XOR gate, and the sum Y_S and carry Y_C are obtained by combining an inverter, transmission gates, and pass transistor type multiplexers. 4T XOR gate produces w0 internally, but Y_S and Y_C output only strong 0 and strong 1.

Fig.6(c) shows a full adder that uses 10 transistors [8]. The input *C* requires in both positive and negative logic, and the output Y_C is obtained in both positive and negative logic. Therefore, it is possible to use the carry Y_C as the *C* input of the next-stage full adder, which is convenient for building a ripple carry adder. w0s or w1s may appear in Y_S and Y_C, and the behavior when Y_S is connected to the input of the next full adder has not been examined.

Fig.6(d) shows a full adder that uses 8 transistors where XOR gates with 3 transistors are employed [9]. In recent technology, the threshold voltage drops as the power supply voltage reduces due to miniaturization and the circuit may not provide the correct output.

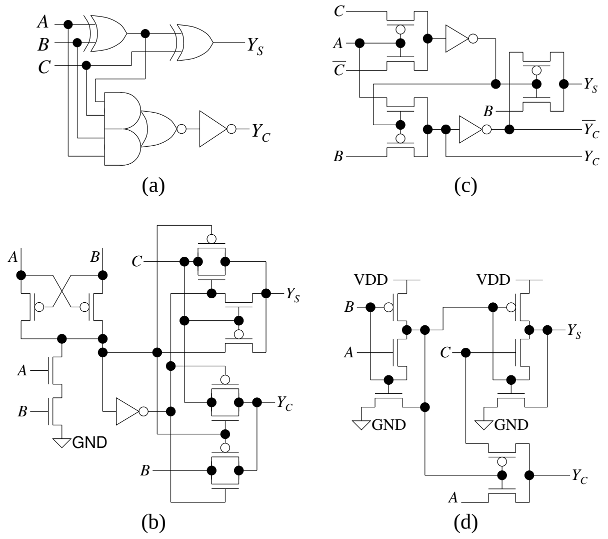


Fig. 6. Full adders. (a) 6T XORs and CMOS gates with 22 transistors. (b) 14 transistors [7]. (c) 10 transistors [8]. (d) 8 transistors [9].

To sum up a large number of input/activation-weight products, it is necessary to connect FAs in multiple stages to perform carry-save addition or tree structure addition. Hence there is a need to devise FAs that have a small number of transistors and can be connected in series.

III. Proposed Method

A. *Efficient summation of 1-bit products*

In the convolution computation, the inputs/activations and weights are multiplied and the products are summed up. In BNN, the product takes either $+1$ (logic 1) or -1 (logic 0). When the kernel size of a convolution is *K*, there are *K* products. Let *P* denote the number of products with the value $+1$. *P* can be calculated by summing up *K* products in logic value. Then the convolution result is obtained as $P-(K-P)=2P-K$. For *M* products, the maximum value of *P* is *M*. An *N*-bit unsigned binary number can represent a value up to 2^N-1. Therefore, choosing $M=2^N-1$ is the most efficient in representing the value of *P* with an *N*-bit binary number. For $N=4$, $M=15$, Fig. 7 shows the circuit that computes 15 1-bit products with XNORs and obtains the value of *P* in a 4-bit binary number. For $K>M$, *K* products are divided into sets with *M* products each, and the circuit is applied to each set to compute the number of $+1$ products in the set and output the value in an *N*-bit binary number. Then those values are summed up to obtain the overall *P* and then the convolution result.

B. *Proposed 11 transistor full adders*

Based on the discussion in Sect. II.C, five types of FA are designed. These five types of FA are named FA1 to FA5, and the circuits are shown in Fig. 8. Fig. 9 shows a product summation circuit using these FAs. FAs are connected in 5 stages in the circuit, and are referred to as Stage 1, Stage 2, and so on in

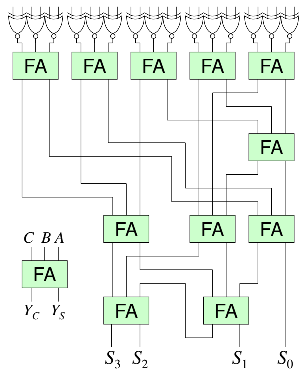


Fig. 7. The number of products of the value '+1' among 15 products is counted and the result is output as a 4-bit unsigned binary number (S_3, S_2, S_1, S_0).

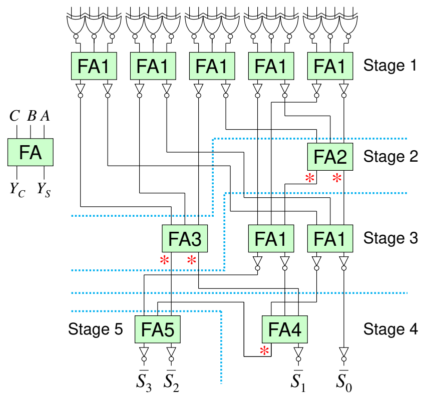


Fig. 9. The proposed circuit for the summation of the products.

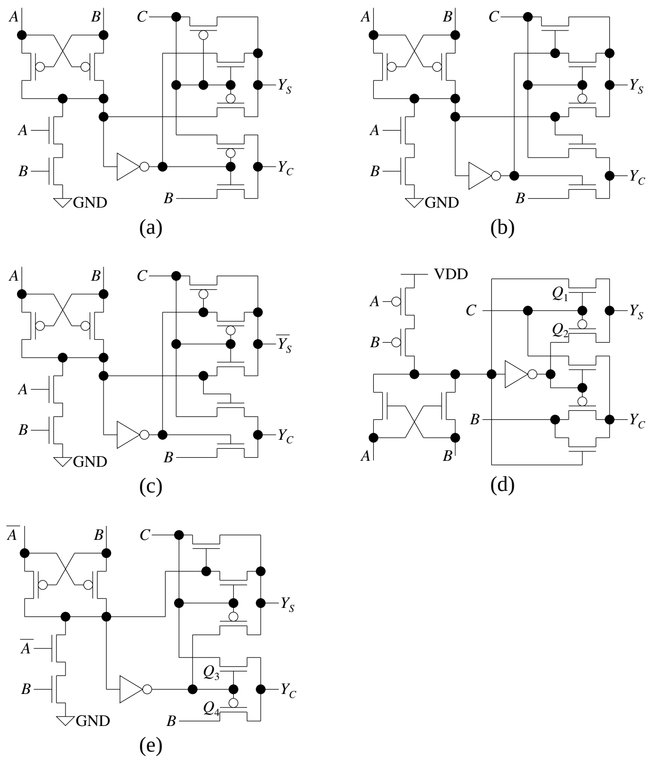


Fig. 8. Designed full adders. (a) FA1, (b) FA2, (c) FA3, (d) FA4, (d) FA5. Note that the output $\overline{Y_S}$ of FA3 and the input $\overline{A}$ of FA5 are inverted.

TABLE II
TRUTH TABLE OF THE PROPOSED FAs

(a) FA1

C	B	A	Y_C	Y_S
0	0	0	0	w0
0	0	1	w0	1
0	0	w1	w0	w1
0	1	0	w0	1
0	w1	0	w0	w1
0	1	1	w1	w0
0	1	w1	w1	w0
0	w1	1	w1	w0
0	w1	w1	w1	w0
1	0	0	0	1
1	0	1	1	0
1	0	w1	1	0
1	1	0	1	0
1	w1	0	1	0
1	1	1	w1	w1
1	1	w1	w1	w1
1	w1	1	w1	w1
1	w1	w1	w1	w1
w1	0	0	0	w1
w1	0	1	w1	0
w1	0	w1	w1	0
w1	1	0	w1	0
w1	w1	0	w1	0
w1	1	1	w1	w1
w1	1	w1	w1	w1
w1	w1	1	w1	w1
w1	w1	w1	w1	w1

(b) FA2

C	B	A	Y_C	Y_S
0	0	0	0	0
0	0	1	0	w1
0	1	0	0	1
0	1	1	w1	0
1	0	0	0	1
1	0	1	w1	0
1	1	0	w1	0
1	1	1	w1	w1

(c) FA3

C	B	A	Y_C	$\overline{Y_S}$
0	0	0	0	1
0	0	1	0	w0
0	1	0	0	w0
0	1	1	w1	1
1	0	0	0	w0
1	0	1	w1	1
1	1	0	w1	1
1	1	1	w1	0

(d) FA4

C	B	A	Y_C	Y_S
0	0	0	0	w0
0	0	w0	0	w0
0	0	1	0	1
0	1	0	0	1
0	1	w0	0	1
0	1	1	1	w0
1	0	0	0	w1
1	0	w0	0	w1
1	0	1	w1	0
1	1	0	w1	0
1	1	w0	w1	w0
1	1	1	1	w1

(e) FA5

C	B	$\overline{A}$	Y_C	Y_S
0	0	1	w0	0
0	0	w1	w0	0
0	0	0	0	1
0	w1	1	0	1
0	w1	w1	0	1
0	w1	0	w1	0
1	0	1	w0	w1
1	0	w1	w0	w1
1	0	0	w1	w0
1	w1	1	w1	0
1	w1	w1	w1	0
1	w1	0	w1	w1

order from the top in Fig. 9. As shown in Fig. 9, the inputs A, B, and C of an FA are arranged from right to left on the upper side of an FA symbol, and the outputs Y_S and Y_C are arranged from right to left on the lower side of an FA symbol.

The products obtained by XNORs are given to FA1 shown in Fig. 8(a) at Stage 1. The value of the product obtained using a 4T XNOR is either 0, 1, or w1. By designing the FA1 not further weaken w1, the product values can be directly input to FA1s, thus eliminating restore inverters between the XNORs and FA1s. The truth table of FA1 is shown in Table II(a). Since the values of FA1 outputs Y_C and Y_S are 0, 1, w0, and w1, it is necessary to restore the signal values before inputting the FA1 output to the full adders in the next stage.

FA2 shown in Fig. 8(b) is used in Stage 2 to add three of the five Y_S which are the output of FA1 of Stage 1 and inverted by the restore inverters. Therefore FA2 receives only 0s and 1s as input values. In order to omit the restoration between the output of FA2 and the full adder in the next stage, FA1 is used in Stage 3, and FA2 is designed to output only w1 in addition to 0 and 1. The truth table of FA2 is shown in Table II(b). The values of the output Y_C and Y_S of FA2 are 0, 1, and w1.

Note that when the three inputs A, B, C of a full adder are represented in negative logic, the outputs Y_C and Y_S are also represented in negative logic. This can be immediately proved by the fact that Y_C is the majority vote of the three inputs and that Y_S is the exclusive OR of the three inputs. Since the FA1

TABLE III
SIMULATION CONDITION

Technology	65 nm CMOS SOTB	
Transistor size	$L_p = L_n = 60$, $W_p = 200$, $W_n = 140$ [nm]	
VDD	0.7 [V]	
Body bias	NMOS Tr	PMOS Tr
Default	0.7 [V]	0.0 [V]
Restore inverter	0.0 [V]	0.7 [V]
XNOR for the product	1.0 [V]	0.0 [V]
Q_1 in FA4, Q_3 in FA5	1.0 [V]	
Q_2 in FA4, Q_4 in FA5		-0.3 [V]

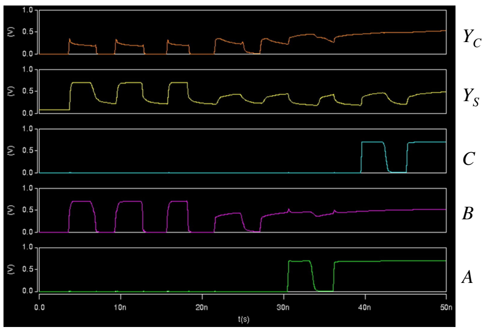


Fig. 10. Circuit simulation result of FA1.

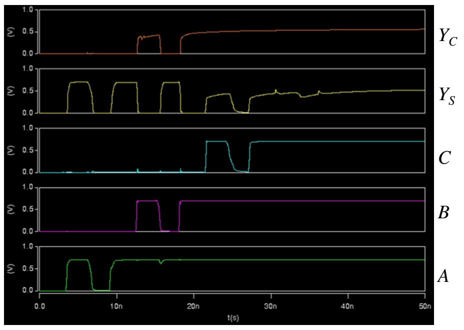


Fig. 11. Circuit simulation result of FA2.

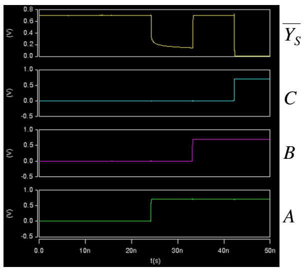


Fig. 12. Circuit simulation result of FA3.

outputs of Stage 1 are inverted by the restore inverter, the inputs of FA2 are given in negative logic and therefore the outputs of FA2 are also represented in negative logic.

The remaining two Y_S (restored and inverted) of the FA1 output of Stage 1 and Y_S of FA2 are given to the FA1 on the right side of Stage 3. The two of the five Y_C of the FA1 outputs of Stage 1 are added by the FA1 on the left side of Stage 3 together with the Y_C of FA2. The remaining three Y_C of the FA1 outputs of Stage 1 are input to the FA3 of Stage 2. FA3 shown in Fig. 8(c) receives only 0s and 1s as input values. In order to omit the restoration between FA3 and the full adders of the next stage which receive the FA3 outputs, the value of the outputs of FA3 is designed to be only one of w0 and w1 in addition to 0 and 1. Since the inputs of FA3 are the outputs of FA1s of Stage 1, they are given in negative logic. The output Y_C of FA3 is in negative logic, while Y_S is designed to be represented in positive logic. This is because the Y_S is an input of the full adder (FA4) in the next stage and the FA4 receives its inputs in positive logic. Hence Y_S output of the FA3 is denoted as $\overline{Y_S}$ to indicate that it is in the logic opposite to A, B, C, and Y_C. The truth table of FA3 is shown in Table II(c). The outputs of the FA1s of Stage 3 are restored with CMOS inverters.

FA4 shown in Fig. 8(d) of Stage 4 receives its input values in positive logic. The A input for FA4 is $\overline{Y_S}$ of FA3, which is either 0, 1, or w0. The values of B and C inputs are 0 or 1. FA4 is designed so that the value of the output Y_C is 0, 1, or w1 and Y_C can be connected to the full adder (FA5) in the next stage without restoration. The truth table of FA4 is shown in Table II(d).

The input A of FA5 shown in Fig. 8(e) of Stage 5 is given in negative logic, and B and C are given in positive logic. The truth table of FA5 is shown in Table II(e).

The outputs of the FA4 and FA5 are restored with inverters and therefore the obtained binary number is represented in negative logic as $(\overline{S_3}, \overline{S_2}, \overline{S_1}, \overline{S_0})$. One more inverter is used in Stage 4 to make the output S_0 in negative logic according to other bits.

Consequently, by designing 5 types of FAs with 11 transistors, each of which devised the appearance of w0s or w1s, restore inverters are eliminated at the output of the XNORs to calculate the products and at the position indicated by asterisks (*) in Fig. 9.

IV. EXPERIMENTAL RESULTS

Circuit simulation was performed for the proposed circuit of the summation of products shown in Fig. 9. The simulation condition is summarized in Table III and the circuit simulator HSPICE was used. The input values to the XNOR gates for the input/activation-weight products were switched at the interval of 3 ns.

Fig. 10 shows the waveforms of the inputs A, B, and C, and the outputs Y_S and Y_C of an FA1 of Stage 1. It is confirmed that Y_S and Y_C were correctly the sum and carry of the given inputs, respectively, and that FA1 receives w1 as inputs and the output value takes 0, 1, w0, and w1, as expected. Fig. 11 shows the waveforms of the inputs and outputs of FA2. Since the inputs of FA2 are restored, A, B, and C do not take the value of w0 and w1. The output takes the value of 0, 1, and w1. Fig. 12 shows the waveforms of the inputs A, B, and C, and the output $\overline{Y_S}$ of FA3. FA3 inputs and outputs values in negative logic except for Y_S. From Fig. 12, it is confirmed that $\overline{Y_S}$, the inversion of Y_S, was correctly obtained and it took the value of 0, 1, and w0. Fig. 13 shows the waveforms of the inputs A, B, and C, and the output Y_C of FA4. The input A took the value of 0, 1, and w0, and Y_C took the value of 0, 1, and w1. Fig. 14 shows the

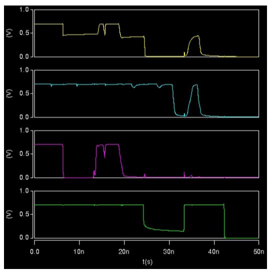


Fig. 13. Circuit simulation result of FA4.

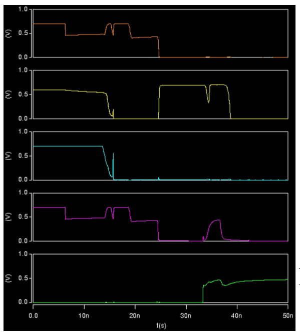


Fig. 14. Circuit simulation result of FA5.

waveforms of the inputs and the output of FA5. FA5 receives the input A in negative logic ($\overline{A}$). From Fig. 14, it is confirmed that the outputs were correctly obtained.

Fig. 15 shows the waveforms of the outputs $\overline{S_3}$, $\overline{S_2}$, $\overline{S_1}$, and $\overline{S_0}$ of the circuit shown in Fig. 9. The input data is given so that the number of XNOR gates that output logic 1 (that is, the product is '+1') is initially 15 and the input data is decremented by 1 every 3 ns until it reaches 0. The result of the summation of products is output as a 4-bit binary number, but it is given in negative logic because of the restore inverters. Therefore, when the number of products of the value 1 is 15, $(\overline{S_3}, \overline{S_2}, \overline{S_1}, \overline{S_0}) = (0000)$, when the number of products is 14, $(\overline{S_3}, \overline{S_2}, \overline{S_1}, \overline{S_0}) = (0001)$, and when the number of products is 0, $(\overline{S_3}, \overline{S_2}, \overline{S_1}, \overline{S_0}) = (1111)$, From Fig.15, it can be confirmed that the proposed circuit operates correctly.

Fig. 16 shows a comparison of the number of transistors between the conventional method and the proposed method. In the conventional method, 15 6T XNOR gates shown in Fig. 2(b) are used to calculate the product, and 11 FAs with 22 transistors shown in Fig. 6(a) are used to calculate the summation. The proposed method is the circuit shown in Fig. 9, which consists of 15 4T XNOR gates shown in Fig. 3(b) for the products, 11 FAs of F1 to F5 with 11 transistors each, and 12 CMOS restore inverters. When these circuits were applied to the convolution calculation of kernel size $3 \times 3 \times 64$, the number of transistors was 14,238 in the conventional method and 9,640 in the proposed method, achieving a reduction of about 32%.

V. Conclusions

An efficient composition of FAs to obtain the summation of input/activation-weight products was proposed. The summation is calculated using FAs with a smaller number of transistors than the conventional FAs to reduce the required total number of transistors for BNNs. Improvement of delay time by transistor sizing and evaluation of dynamic and static power consumption are future tasks.

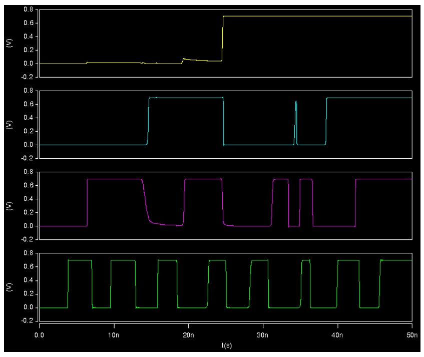


Fig. 15. The number of products of the value '+1' is counted and the result is output with a 4-bit unsigned binary number $(\overline{S_3}, \overline{S_2}, \overline{S_1}, \overline{S_0})$.

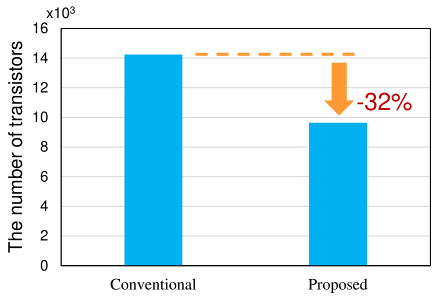


Fig. 16. The number of transistors of the conventional and the proposed designs.

References

[1] Convolutional Neural Networks for Visual Recognition, Stanford University, http://cs231n.github.io/

[2] Y. Umuroglu, et al., "FINN: A Framework for Fast, Scalable Binarized Neural Network Inference," Proc ISFPGA 2017, pp. 65–74, 2017.

[3] F. Conti, et al., "XNOR Neural Engine: A Hardware Accelerator IP for 21.6-fJ/op Binary Neural Network Inference," IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol. 37, no. 11, pp. 2940–2951, 2018.

[4] C. Chi, J. R. Jiang, "Logic Synthesis of Binarized Neural Networks for Efficient Circuit Implementation," Proc. ICCAD 2018, pp. 1–7, 2018.

[5] R. Sugimoto, N. Ishiura, "Parameter Embedding for Efficient FPGA Implementation of Binarized Neural Networks," Proc. SASIMI 2019, pp. 41–45, 2019.

[6] J. M. Wang, et al., "New Efficient Designs for XOR and XNOR Functions on the Transistor Level," IEEE Journal of Solid-State Circuits, vol. 29, no. 7, pp. 780–786, 1994.

[7] E. Abu-Shams, M. Bayoumi, "A New Cell for Low Power Adders," Proc. Int. Midwest Symp. Circuits Syst. 1995, pp. 1014–1017, 1995.

[8] J.-F. Lin, et al., "A Novel High-Speed and Energy Efficient 10-Transistor Full Adder Design," IEEE Trans. Circuits and Systems I: Regular Papers, vol. 54, no. 5, pp. 1050–1059, 2007.

[9] M. Kumar, et al., "Single Bit Full Adder Design using 8 Transistors with Novel 3 Transistors XNOR Gate," Proc. Int. Journal of VLSI Design and Communication Systems, vol. 2, no. 4, pp. 47–59, 2012.