Long Short-Term Memory Network Design for Analog Computing

ZHOU ZHAO, ASHOK SRIVASTAVA, LU PENG, and QING CHEN, Louisiana State University

We present an analog-integrated circuit implementation of long short-term memory network, which is compatible with digital CMOS technology. We have used multiple-input floating gate MOSFETs as both the front-end to obtain converted analog signals and the differential pairs in proposed analog multipliers. Analog crossbar is built by the analog multiplier processing matrix and bitwise multiplications. We have shown that using current signals as internal transmission signals can largely reduce computation delay, compared to the digital implementation. We also have introduced analog blocks to work as activation functions for the algorithm. In the back-end of our design, we have used current comparators to achieve the output to be readable to external digital systems. We have designed the LSTM network with the matrix size of 16×16 in TSMC 180nm CMOS technology. The post-layout simulations show that the latency of one computing cycle is 1.19ns without memory, and power dissipation of the single analog LSTM computing core with 2 kilobytes SRAM at 200MHz is 460.3mW. The overhead of power dissipation due to SRAM access is 8.3%, in which the computing of each LSTM layer requires one computing cycle. The energy efficiency is 0.95TOP/s/W.

CCS Concepts: • **Hardware → Application specific integrated circuits**; *Full-custom circuits*;

Additional Key Words and Phrases: Long short-term memory, analog signal processing, multiple-input floating gate MOSFET

ACM Reference format:
Zhou Zhao, Ashok Srivastava, Lu Peng, and Qing Chen. 2019. Long Short-Term Memory Network Design for Analog Computing. *J. Emerg. Technol. Comput. Syst.* 15, 1, Article 13 (January 2019), 27 pages.
http://dx.doi.org/10.1145/3289393

1 INTRODUCTION

Algorithms and circuit designs that support neural computation are the subjects of intense research (Stewart and Eliasmith 2014; Wang et al. 2014; Wang et al. 2016; Taigman et al. 2014). Recurrent neural network (RNN) is one of the artificial neural networks (ANN) that can process real-time data (Pearlmutter 1989). As an extension of various RNNs, long short-term memory (LSTM) is highly desirable to avoid the long-term dependency problem (Pulver and Lyu 2017; Hochreiter and Schmidhuber 1997). In hardware implementation of LSTM, the involved matrix multiplication and non-linear activation functions take the most time, and related blocks consume the most power, compared to the rest of blocks in LSTM. The challenge of combining the algorithm and the

This work is supported by the National Science Foundation grant No. 1422408.
Authors' addresses: Z. Zhao, A. Srivastava (corresponding author), L. Peng, Division of Electrical and Computer Engineering, Louisiana State University, Baton Rouge, LA 70803; Q. Chen, Division of Computer Science and Engineering, Louisiana State University, Baton Rouge, LA 70803; email: {zzhao13, eesriv, lpeng, qchen11}@lsu.edu.

1550-4832/2019/01-ART13 $15.00
http://dx.doi.org/10.1145/3289393

corresponding design implementation depends on how to efficiently address matrix multiplication of large size, and obtaining accurately the output through activation functions. Current work on VLSI implementation of neural computation has been focused mainly on convolution neural network (CNN) (Ni et al. 2017; Moons et al. 2017; Sim et al. 2016; Chen et al. 2016) in digital design. The hardware implementations of LSTM mostly use FPGA (Guan et al. 2017a; Guan et al. 2017b; Ferreira and Fonseca 2016). The computation speed of LSTM in FPGA-based implementations is mainly restricted by digital gates operating at ~100MHz global clock, which is lower than that in ASIC design. Shin et al. (2017) presented a neural processor that can implement LSTM in deep sub-micron CMOS technology. In this work, both lookup table (LUT) and register array are used in advance to store all possible results of product and quantization codes to avoid large latency introduced by matrix multiplication and activation functions.

Recently, analog signal processing (ASP) has emerged for neural computation. Neuron-like analog cells have been reported to implement neural computation (Bong et al. 2017; Maliuk and Makris 2015; Corradi et al. 2014; Maliuk and Makris 2012). Amant et al. (2014) proposed an analog computation method to make ASP work for complex neural networks reliably. However, the signal interface implemented by digital-to-analog converter (DAC) and analog-to-digital converter (ADC) increases circuit area and power dissipation. The large overhead of power dissipation due to both DACs and ADCs working for ASP is also seen in the work of Shafiee et al. (2016), in which the internal ASP can largely accelerate CNN computing at the cost of over half of the entire power dissipated by signal interface. The interface circuit also restricts the maximum frequency due to tradeoff between effective number of bits (ENOB) and sample rate in ADC. If we do not take signal interface into consideration and only compare ASP and DSP in addressing complex neural computation, then ASP has several advantages over DSP: (a) A single analog signal can represent the digital signal with multiple bits; (b) Current neural algorithms involve many non-linear activation functions, which are easier to implement in analog than in digital circuits; c) Matrix multiplication (which is widely used in neural computations) implemented in analog design is much faster than in digital design. However, DACs and ADCs connecting external blocks add real estate to the chip and computation latency.

In the past, analog ICs implemented with multiple-input floating gate (MIFG) MOSFETs simulating biological neurons have been reported (Holler et al. 1989; Shimabukuro et al. 1991; Aslam-Siddiqi et al. 1998). In the work of Holler et al. (1989), the neuron-based analog circuit gives the variable gain to obtain the desired performance. However, the feedback loop increases the computation latency. The work in Shimabukuro et al. (1991) shows a novel current-mode analog neuron-like cell with floating gate MOSFETs fabricated in silicon on sapphire (SOS) CMOS technology. The output swing is very small, which is not suitable in multi-bit applications. The work in Aslam-Siddiqi et al. (1998) has used switch-capacitor (SC) circuits using floating gate MOSFETs to build a programmable neuron circuit. The output swing is large and can be used for multi-bit computations. However, the voltage-mode process reduces the computation latency and the SC circuit requires a high-accuracy clock block under high-frequency operation. In this work, we propose an analog VLSI implementation of LSTM without DAC or ADC. The main work in this article is presented as follows: (a) We have analyzed the algorithm of LSTM and modified it to be used in our implementation based on analog IC design; (b) Compared to traditional data conversion costing high-analog IC, we have proposed an energy-efficient and simple method of data conversion that uses MIFG MOSFETs simulating biological neurons at the front-end to obtain analog signals from digital input, and current comparators at the back-end to achieve ADC function. This method makes the designed system fully compatible with the digital environment. The LSTM network includes a crossbar network built by analog multiplication cells, current/voltage converters, and analog cells to implement sigmoid and hyperbolic tangent

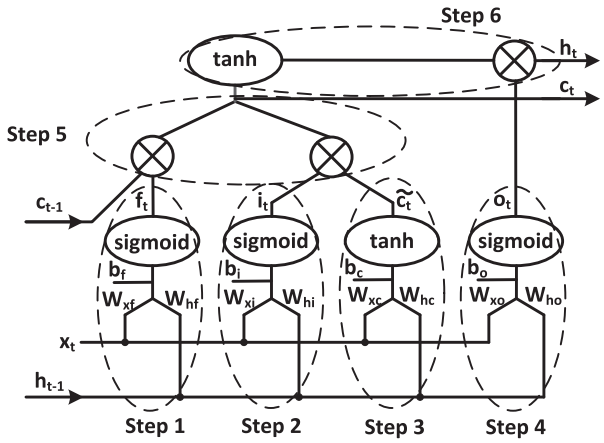


Fig. 1. Standard cell of LSTM network.

functions. In internal operation of LSTM, we have mainly used the current carry signal as the transmission signal; (c) Physical design of the LSTM computing core with 16 × 16 matrix is done in TSMC 180nm CMOS technology and the results of core are presented.

The article is organized as follows: Section 2 and Section 3 present description of LSTM and MIFG MOSFETs, respectively; Section 4 presents the analog implementation of LSTM; Section 5 presents simulation results and further discussion. Section 6 summarizes the work.

2 LONG SHORT-TERM MEMORY NETWORK

RNN is a cycle network that allows data processing with the dependency of past information, and LSTM is one of the extensions of RNN. When the time span between two instances is large, the network will lose the ability of connecting two points used for processing and prediction. LSTM can solve the problem of long-term dependency, and so it can be used in a variety of applications (Hochreiter and Schmidhuber 1997). LSTM uses the pipelined method, utilizing the same standard cells. The output of current state influences the next state. The standard cell of LSTM is shown in Figure 1, which consists of six independent steps to follow dotted ellipses.

The first four steps in Figure 1 represent forget gate layer, input gate layer, cell gate layer, and output gate layer, respectively. In these, the argument of each activation function is the sum of two matrix multiplications and a bias, $\boldsymbol{W_x\, x_t + W_h h_{t-1} + b}$, where $\boldsymbol{W_x}$, $\boldsymbol{x_t}$, $\boldsymbol{W_h}$, $\boldsymbol{h_{t-1}}$, and $\boldsymbol{b}$ are weighted matrix of input vector, input vector; weighted matrix of output vector, output vector; and bias, respectively. Considering the analog implementation of LSTM, there are two additions of column vectors in a single argument going through an activation function. To simplify the circuit design, we can convert these two additions into a single addition. If the dimensions of all weighted matrices are $N \times N$ and the dimensions of the rest of column vectors are $N \times 1$ in LSTM, then we can embed the last bias term into the first term in the argument of the first four steps. Based on the Equations (3)–(8) in the work of Guan et al. (2017b), the new form of LSTM can then be expressed as follows:

$$\begin{cases} f_t = sigmoid(\widehat{W_{xf}}\widehat{x_t} + \widehat{W_{hf}}\widehat{h_{t-1}}) \\ i_t = sigmoid(\widehat{W_{xl}}\widehat{x_t} + \widehat{W_{hl}}\widehat{h_{t-1}}) \\ \widetilde{c_t} = tanh(\widehat{W_{xc}}\widehat{x_t} + \widehat{W_{hc}}\widehat{h_{t-1}}) \\ o_t = sigmoid(\widehat{W_{xo}}\widehat{x_t} + \widehat{W_{ho}}\widehat{h_{t-1}}) \\ c_t = f_t \circ c_{t-1} + i_t \circ \widetilde{c_t} \\ h_t = o_t \circ tanh(c_t) \end{cases}, \tag{1}$$

In Equation set (1), each new weighted matrix of the input vector $\hat{W}_x$ represents $[W_x, b]$, each new input vector $\hat{x}_t$ represents $[x_t^T, 1]^T$, each new weighted matrix of the output vector $\hat{W}_h$ represents $[W_h, 0]$, and each new output vector $\hat{h}_{t-1}$ represents $[h_{t-1}^T, 0]^T$. From Equation set (1), we can see that all bias terms are integrated in $\hat{W}_x\hat{x}_t$ and there is only one addition in the argument of the first four layers. The dimensions of all new weighted matrices and column vectors in the arguments are $N \times (N+1)$ and $(N+1) \times 1$, respectively. It should be noted that in analog implementation, each element in both $\hat{W}_x\hat{x}_t$ and $\hat{W}_h\hat{h}_{t-1}$ is an analog signal. The matrix multiplication corresponding to $\hat{W}_x\hat{x}_t$ is $N \times (N+1)$ matrix multiplied by $(N+1) \times 1$ column vector to cover the addition with the bias term. The matrix multiplication corresponding to $\hat{W}_h\hat{h}_{t-1}$ can be simplified to $N \times N$ matrix multiplied by $N \times 1$ column vector without the null multiplication. Through this new form of LSTM shown in Equation set (1), we have removed one stage implementing the addition with the bias term to not only accelerate computation, but also to reduce both the circuit cost and interconnection complexity. The output in every sub-equation is still an $N \times 1$ column vector as desired. All elements in matrices still do the same calculation. Thus, there is no error introduced by the proposed modification. The computation flow as shown in Equation set (1) includes matrix multiplication, bitwise multiplication to obtain Hadamard product, addition, and two activation functions, which are sigmoid and hyperbolic tangent. The first four steps can be computed simultaneously. Section 4 will further introduce the circuit simplification to speed up the entire computation flow through both specific data conversion and physical interconnection.

3 MULTIPLE-INPUT FLOATING GATE MOSFET

Traditional transistors have only one input connecting to the gate to control the current transport in the channel (Nicollian and Brews 1982). In analog IC design, normal MOSFETs used in the input stage only map injective functions to the output. To overcome this restriction, MIFG MOSFET is introduced to increase both the inputs for a single transistor and the reconfigurable output (Shibata and Ohmi 1992; Srivastava and Srinivasan 2002; Ramirez-Angulo et al. 2004; Subramanian 2005). The MIFG MOSFET has two gate layers. The bottom gate layer is a floating gate that is over the thin oxide of the transistor channel; and the top gate layer, which is over the floating gate and isolated by another thin oxide, is to achieve the programmable input via multiple gate inputs. The Equivalent capacitor-based circuit and a 3D view of a MIFG MOSFET are shown in Figures 2(a) and 2(b), respectively. The surface potential on floating gate Φ_f decides turning on and off the transistor and is expressed as follows (Venkata 2002):

$$\emptyset_f = \frac{C_1V_1 + C_2V_2 + \cdots + C_nV_n}{C_{ox} + C_1 + C_2 + \cdots C_n + C_p}, \tag{2}$$

where $V_1, V_2, \ldots, V_n$ represent multiple input voltages that are inputs to the top gate layer. Their corresponding coupling capacitors are $C_1, C_2, \ldots, C_n$, all of which are contributed by the oxide layer isolating the floating gate and each top input gate. C_{ox} and C_p are the capacitors between the floating gate and the substrate, and parasite capacitor, respectively. C_p can be safely removed, since its value is much smaller than the rest of capacitors. When the dimension of a MIFG MOSFET is small, we can neglect C_{ox}. While in analog IC design using MIFG MOSFETs, dimensions of some transistors may be relatively large to assure that all transistors work in their correct regions of operation, C_{ox} should be taken into consideration. Normally, one effective method to suppress the non-linearity from C_p and C_{ox} is to set $C_1, C_2, \ldots, C_n$ to relatively large values. For the physical layout design of MIFGs using the concept of Figure 2(b), a double-poly layer is used to form a floating gate and coupling capacitive gates. Thus, we observe that this style of physical design can cause the values of coupling capacitors to be restricted by the dimension of the transistor connected with MIFGs, which reduces the design freedom of analog IC. To avoid the mutual restriction between coupling capacitors and customized ratio of the transistor with MIFGs in this work, for any

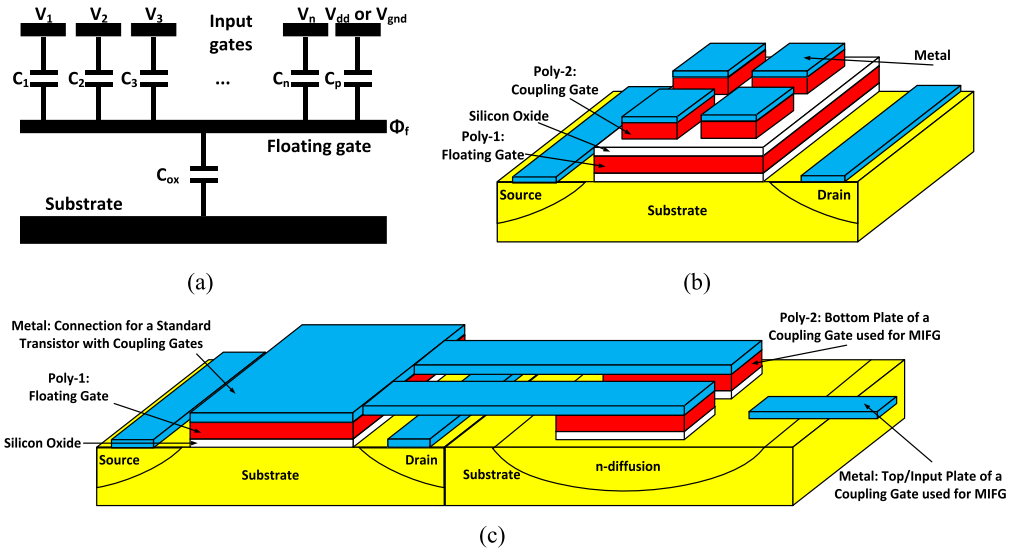


Fig. 2. MIFG MOSFET: (a) equivalent capacitor-based circuit, (b) 3D view of overlapping-dual-poly structure, and (c) 3D view of adjacent n-diffusion capacitor structure. Note that both poly and active contacts are ignored in two 3D views.

transistor with MIFGs, we have used another structure shown in Figure 2(c) to achieve coupling capacitors, which are implemented by standard n-diffusion capacitors. This design style of MIFG MOSFETs is fully compatible with the fabrication flow of standard CMOS and already proposed, fabricated, and tested in the work of Shibata and Ohmi (1992), Srivastava and Srinivasan (2002), Srivastava and Venkata (2003), and Subramanian (2005). The essence of this structure is that the coupling capacitors are designed in the area beside MIFG transistors but not over the floating gate. Through this structure, the dimension of a MIFG transistor and values of coupling capacitors are independent of each other. Another important issue is the method of simulating MIFG device and circuit. In this work, we use the resistor-capacitor pairs with a large grounding resistor to achieve signal synchronization and solve the "floating node" error which is due to multiple inputs with the same electrical node in SPICE simulation (Sánchez-Sinencio 2010).

3.1 MIFG MOSFETs for DAC Design

The property of a MIFG MOSFET through which the output can be determined by multiple inputs, matches to the signal flow in DAC, which is the initial stage used for ASP. Therefore, we can use MIFG MOSFETs to achieve a DAC function. For N-bit digital signal applied in a MIFG MOSFET, $V_{D\_N\text{-}1}$, $V_{D\_N\text{-}2}$, $\ldots V_{D\_1}$, V_{D_0}, if we set the ratio of their corresponding coupling capacitors to $2^{N\text{-}1}$:$2^{N\text{-}2}$: $\ldots$:2^1:2^0, then the surface potential on the floating gate of a MIFG MOSFET can be programmed as follows:

$$\emptyset_f = \frac{2^{N-1}C_{unit}V_{D\_N-1} + 2^{N-2}C_{unit}V_{D\_N-2} + \cdots + 2^0 C_{unit}V_{D\_0}}{C_{ox} + 2^{N-1}C_{unit} + 2^{N-2}C_{unit} + \cdots + 2^0 C_{unit} + C_p}, \tag{3}$$

where C_{unit} is the unit capacitor implemented in physical design. If we neglect both C_{ox} and C_p, then the numerator in Equation (3) shows a function of DAC in which the multiple inputs and the surface potential are digital and analog signals, respectively. If the denominator in Equation (3) is C_{total}, then the step voltage for the conversion is $V_{DD} \times C_{unit}/C_{total}$. This implementation does not require an independent circuit block to achieve a DAC function. Input transistors in analog

multipliers of this work are directly connected to multiple digital inputs. Traditional DACs in ASP systems need several feedback loops with operational amplifiers to achieve stability, the required response time will largely increase the conversion latency. However, the proposed one, in which the multiple inputs directly couple to the floating gate of a transistor, does not need response time or the consideration of stability, and thus is much faster than traditional DACs. The output of the proposed conversion is directly applied to the channel of the transistor to control its state to achieve multiplication. In addition, the power dissipation of the proposed conversion is relatively smaller than that in traditional DAC due to the cancellation of feedback loop. To increase the precision in computation, it is feasible to increase the C_{unit} in Equation (3) at the cost of the area due to coupling capacitors. In Section 4.1, the design of analog multiplier with the proposed DAC function using MIFG MOSFETs will be described.

4 LONG SHORT-MEMORY NETWORK DESIGN

4.1 Analog Multiplier with Digital Signal Input

Analog multipliers are more suitable to process data with increasing bit length than digital multipliers (Kim and Park 1987; Song and Kim 1990). However, it is very difficult to achieve rail-to-rail input and output voltage swings. One feasible method to increase the signal swing of an analog multiplier is to increase supply voltage at the cost of power dissipation (Babanezhad and Temes 1985; Soo and Meyer 1982). Another method is to allow transistors in analog multipliers to work in the linear region (Han and Sinencio 1998). To save power, we scaled down the swing of input signals. This is needed to make input digital signals and surface potential on the floating gate after data conversion compatible with the signal flow without increasing the supplied voltage. For the input stage of an analog multiplier using MIFG MOSFETs, we introduced another coupling capacitor, C_g connecting to the ground. Equation (3) can be expressed as follows:

$$\emptyset_f = \frac{2^{N-1}C_{unit}V_{D\_N-1} + 2^{N-2}C_{unit}V_{D\_N-2} + \cdots + 2^0 C_{unit}V_{D\_0}}{C_{ox} + 2^{N-1}C_{unit} + 2^{N-2}C_{unit} + \cdots + 2^0 C_{unit} + C_p + C_g}, \tag{4}$$

Setting the allowable input swing to $[0, \alpha V_{dd}]$, when all digital inputs are high logic, the surface potential reaches to the upper bound, αV_{dd}. This case can be expressed as follows:

$$\alpha V_{dd} = \frac{2^{N-1}C_{unit}V_{dd} + 2^{N-2}C_{unit}V_{dd} + \cdots + 2^0 C_{unit}V_{dd}}{C_{ox} + 2^{N-1}C_{unit} + 2^{N-2}C_{unit} + \cdots + 2^0 C_{unit} + C_p + C_g}, \tag{5}$$

Since C_{ox} and C_p are small enough compared to other coupling capacitors, we can neglect these two terms. Then, Equation (5) can be rewritten as the function of C_g as follows:

$$C_g = \frac{1-\alpha}{\alpha}(2^{N-1} + 2^{N-2} + \cdots + 2^1 + 2^0)C_{unit}, \tag{6}$$

Depending on Equation (6), we can use C_g with other coupling capacitors shown in Equation (4) to execute the function of DAC with the scaling down of the input voltage.

Based on the design presented in the work of Song and Kim (1990), we propose an analog multiplier with programmable MIFG MOSFETs guided by Equations (4) and (6) as shown in Figure 3. It is a differential input pair mapping to differential output in four-quadrant. The two input pairs generate the two square terms, $[(V_{1+}\text{-}V_{1-})+(V_{2+}\text{-}V_{2-})]^2$ and $[(V_{1+}\text{-}V_{1-})\text{-}(V_{2+}\text{-}V_{2-})]^2$. Then, through the cross-interconnection of source followers with inputs, these two terms are subtracted with each other so that the internal square terms such as $(V_{1+}\text{-}V_{1-})^2$ and $(V_{2+}\text{-}V_{2-})^2$ are cancelled and only the term with the function of $(V_{1+}\text{-}V_{1-})(V_{2+}\text{-}V_{2-})$ is left. The voltage output can be expressed as follows:

$$V_{o+} - V_{o-} = R_{load}(I_{o+} - I_{o-}) = \frac{1}{2}K_n\left(\frac{W}{L}\right)_{sf} R_{load}[(V_{1+} - V_{1-})(V_{2+} - V_{2-})], \tag{7}$$

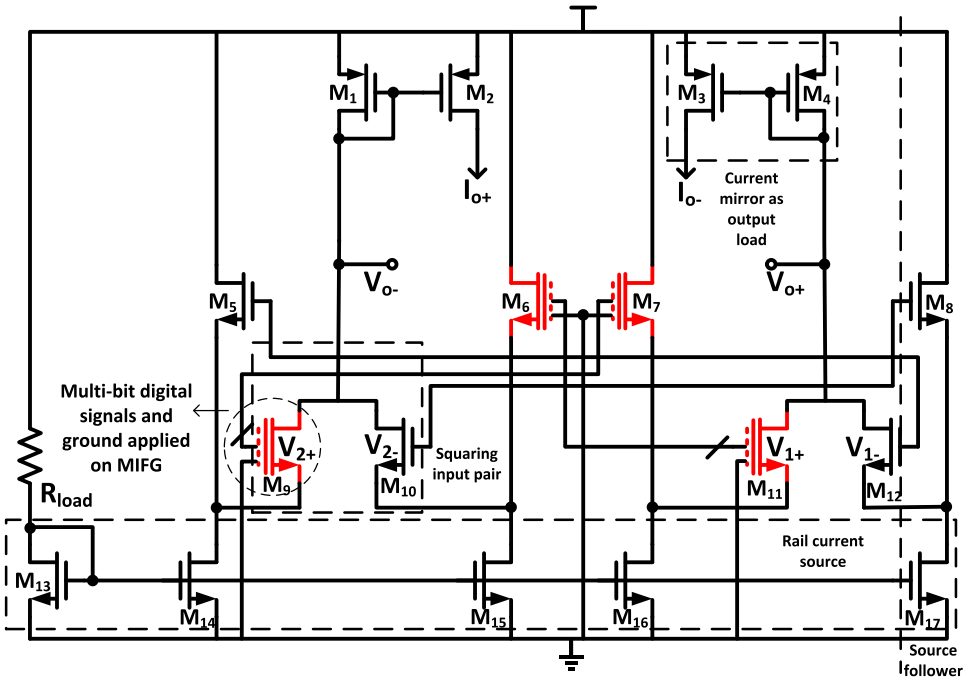


Fig. 3. Circuit diagram of the proposed analog multiplier with the functions of both voltage scaling and DAC. Highlighted transistors are MIFG MOSFETs.

where K_n is the transconductance parameter of n-channel transistors and R_{load} is the resistance of diode-connected transistors (M_1 and M_4) in the output load. $(W/L)_{sf}$ represents the dimension ratio of the transistors (M_5, M_6, M_7, and M_8) in source followers. Looking into Equation (7), K_n and $(W/L)_{sf}$ are determined by the fabrication process and the unit gain of the source follower, respectively. R_{load} in the circuit is sensitive to temperature with a positive temperature coefficient (Allen and Holberg 2002). If we use the current signal as the output, then R_{load} in Equation (7) can be cancelled to mitigate the signal variation due to the temperature fluctuation. The most significant advantage of using current signal as the transmission signal is that the addition of two current signals can be achieved simply by the interconnection using Kirchhoff's current law. However, the addition of two voltage signals cannot be achieved by direct series connection and results in a voltage sum with non-linear composition (Mead and Mohammed 2012). To obtain an accurate sum of two voltage signals, the mixer is required at the cost of delay and power dissipation. Therefore, considering the algorithm of LSTM and the hardware cost, it is better to use current signal as the transmission signal than the voltage signal.

The feasibility of quantization used in LSTM to reduce hardware cost has been studied by Shin et al. (2017) and Han et al. (2016). In our work, we set the single number in both matrices and column vectors to be a 4-bit digital signal and α in Equation (5) to 0.5, which means the input voltage is scaled down from 0V to 0.9V,which is half of the supplied voltage (1.8V) in the TSMC 180nm CMOS process. Combining the analysis from Equations (4), (6), and (7), the output current signal can be expressed as follows:

$$I_{o+} - I_{o-} = \frac{1}{4} K_n \left(\frac{W}{L}\right)_{sf} \gamma_{(W/L)} (V_1 V_2)$$

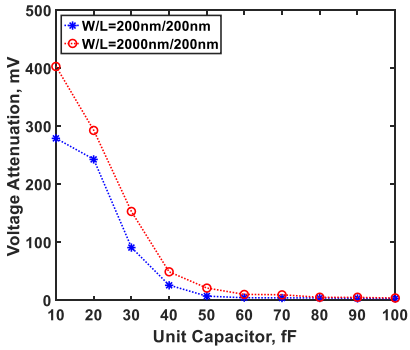


Fig. 4. Voltage attenuation versus unit capacitor in MIFG-driven input transistor.

Table 1. W/L Ratio of Transistors in the Proposed Analog Multiplier using MIFG MOSFETs of Figure 3

Transistor	W/L (nm/nm)
$M_{1\text{-}4}$	10,000/400
$M_{5\text{-}12}$	200/200
M_{13}	400/400
$M_{14\text{-}17}$	8,000/400

$$= \frac{1}{4}K_n\left(\frac{W}{L}\right)_{sf}\gamma_{(W/L)}\left(\frac{8C_{unit}V_{A_3}+4C_{unit}V_{A_2}+2C_{unit}V_{A_1}+C_{unit}V_{A_0}}{8C_{unit}+4C_{unit}+2C_{unit}+C_{unit}+15C_{unit}}\right) \times\left(\frac{8C_{unit}V_{B\_3}+4C_{unit}V_{B\_2}+2C_{unit}V_{B\_1}+C_{unit}V_{B\_0}}{8C_{unit}+4C_{unit}+2C_{unit}+C_{unit}+15C_{unit}}\right), \quad (8)$$

where $\gamma_{(W/L)}$ is the dimension ratio of two p-type MOSFETs in the current mirror used for the output stage. The last two terms in brackets are two 4-bit digital signals to be multiplied. The ratio of four coupling capacitors used in the function of DAC is 8:4:2:1, as shown in Equation (8). C_g is $15C_{unit}$, which is used for signal scaling. Using Equation (8), the external digital signals can be multiplied.

For determining the value of C_{unit}, we need to study the signal attenuation at the surface potential due to the influence of C_{ox} and C_p. Setting all digital inputs to high logic, Figure 4 shows the signal attenuation dependence on varying C_{unit}. We can see that when the dimension of the driven transistor is large (W/L = 2,000nm/200nm), it requires a large-unit capacitor to mitigate the signal attenuation at the surface potential, since C_{ox} is proportional to the dimension of a transistor. When the coupling capacitor is larger than that of C_p and C_{ox}, the surface potential can accurately reflect the programmable input using MIFG MOSFETs. We should consider circuit area also, because capacitors in MIFG MOSFETs occupy a larger area than in normal transistors. Therefore, we choose 50fF as the unit-size capacitor as a tradeoff between area and signal accuracy.

Table 1 shows W/L ratios of transistors shown in Figure 3. Setting the resistor used for biasing the tail current to 5kΩ and common mode voltage to 0.9V, we obtain the transient simulation of the multiplier as shown in Figure 5(a). It can be seen that the proposed multiplier can process the signal from standard digital interface, 4-bit digital signal (V_{pp} = 1.8V) to analog current signal with the swing of ±180μA. Figure 5(b) is the dc transfer curve to observe the linearity of the proposed multiplier. We observe that a smaller input leads to a better linearity of the output. Four curves can

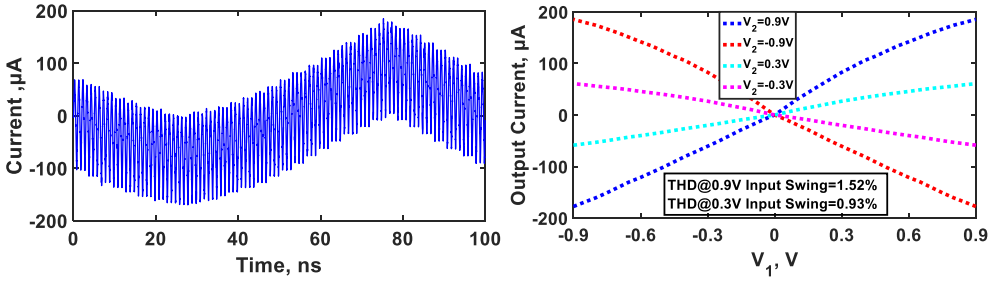


Fig. 5. The proposed analog multiplier with MIFGs: (a) transient simulation, two 4-bit inputs are from all high logic (1.8V), flipping to all low logic (0V) periodically under 1GHz and 10MHz and (b) dc transfer curve.

be roughly seen as linear curves that ensure the correct multiplication. Total harmonic distortion (THD) is another important metric to evaluate the performance of analog multipliers. THDs with input of 0.9V and 0.3V swings are 1.52% and 0.93%, respectively; both are acceptable and so cannot largely influence the multiplication due to harmonics. This analog multiplier will also be used for both matrix multiplication and bitwise multiplication, which will be introduced later. Compared to digital multiplier with multiple bits, the proposed method can implement both multiplication and DAC function as the front-end of digital interface. The most attractive feature is that the analog multiplier is much faster than the digital multiplier, since it uses only a single analog stage to obtain the result instead of the pipelined topology in digital multiplication.

4.2 Design of Other Functional Blocks

Besides analog multipliers used for matrix and bitwise multiplications, there are circuits implementing activation functions, current comparator, and current mirror required for the entire computation flow of LSTM. The digital implementation of an activation function usually uses LUT to search results or a step linear function to approximate the non-linear curve (Shin et al. 2017; Piazza et al. 1993). The accuracy mainly relies on table capacity and search principle. In analog design of an activation function, only one circuit block can execute the computation (Al-Ruwaihi 1997; Tabarce et al. 2005). In our design, the received signal from multiplication is a current signal, and the output signal from an activation function is used for the following bitwise multiplication, according to LSTM algorithm. Thus, we need to build a block of the activation function with the current signal as an input and voltage signal as an output. The analog circuit design of sigmoid and hyperbolic tangent functions is shown in Figures 6(a) and 7(a), respectively. The intermediate block computing the activation function uses voltage signal as the input and current signal as the output. It can be seen that there are two current/voltage converters to match to the correct signal flow. Since the hyperbolic tangent function has a negative value, a reverse-voltage bias is used to allow the output to drop to a negative value. Using the current signal as an input, the transfer functions corresponding to these two circuits are expressed as follows:

$$\begin{cases} V_{sigmoid\_out}(I_{in}) = \frac{1}{2}K_p\left(\frac{W}{L}\right)_5\left(g_{m6}r_{o2}r_{o4}I_{in} - \sqrt{\frac{g_{m6}^2}{K_nK_p\left(\frac{W}{L}\right)_2\left(\frac{W}{L}\right)_6}} - V_{dd} - 2V_{tp}\right)^2 r_{o10} \\ V_{\tanh\_out}(I_{in}) = \left\{\frac{1}{2}K_p\left(\frac{W}{L}\right)_5\left(g_{m6}r_{o2}r_{o4}I_{in} - \sqrt{\frac{g_{m6}^2}{K_nK_p\left(\frac{W}{L}\right)_2\left(\frac{W}{L}\right)_6}} - V_{dd} - 2V_{tp}\right)^2 - I_{11/12}\right\} r_{o10} \end{cases}, \tag{9}$$

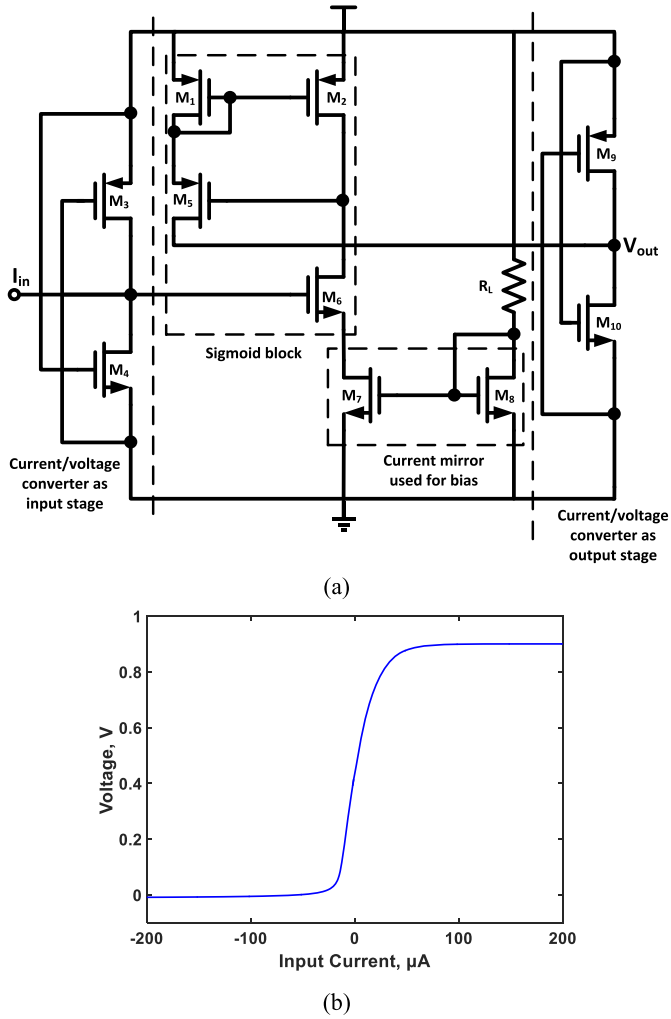


Fig. 6. Analog implementation of sigmoid function: (a) circuit diagram and (b) transfer characteristics.

In the transfer function of sigmoid, the input current shows a square relation with output voltage in effective input swing. Once the input current is out of effective input swing after front-end I/V converter (M_3 and M_4) due to M_5 as a feedback controlling the common source (CS) amplifier (M_2 and M_6), the output voltage will stop changing to go to cut-off region and form top and bottom signal lines. This principle is also applied in the transfer curve of hyperbolic tangent function. However, we have introduced a reverse voltage bias (M_{11} and M_{12}) to pull down the swing of output voltage in hyperbolic tangent function. The reason we let two transistors work in sub-threshold region to generate the negative voltage bias is that the characteristic of a transistor in sub-threshold region is less sensitive to external voltage variation than it is in linear or saturation region. The *W/L* ratio of transistors in both circuits is listed in Table 2. R_L in Figure 6(a) is 5kΩ, R_{L1} in Figure 7(a) is 20kΩ, and R_{L2} in Figure 7(a) is 5kΩ. Figure 6(b) and Figure 7(b) show the simulated static characteristics of sigmoid and hyperbolic tangent functions, respectively.

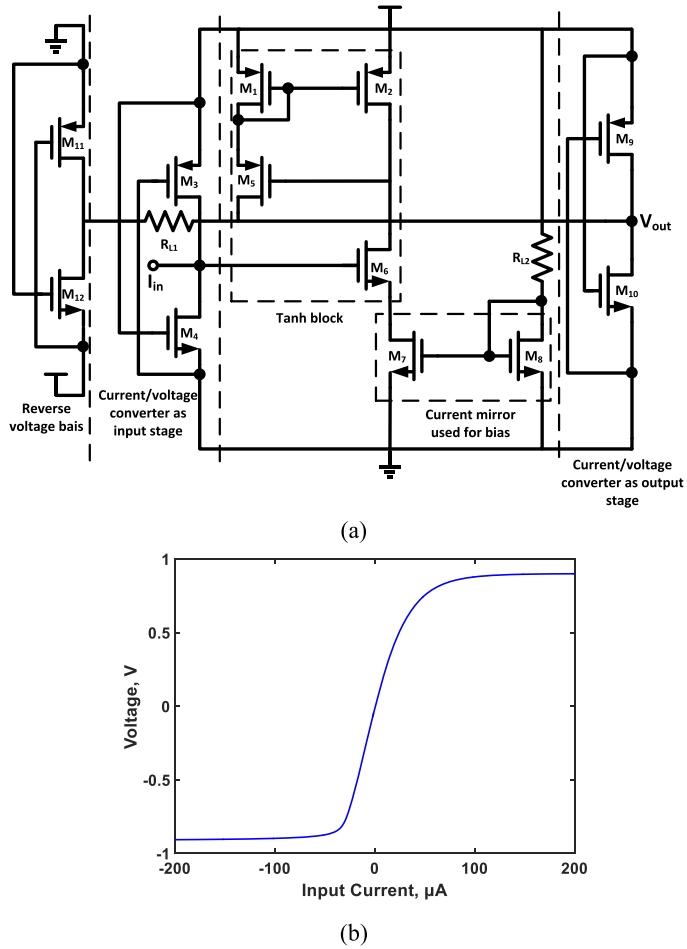
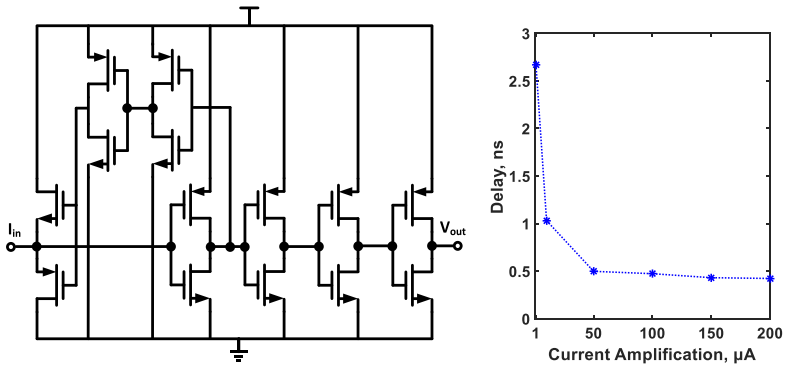


Fig. 7. Analog implementation of hyperbolic tangent function: (a) circuit diagram and (b) transfer characteristics.

For the back-end of our design, in which the analog current signal is to be converted to a digital voltage signal, we use current comparators for the signal conversion. The delay of a current compactor is sensitive to the current resolution (Tang and Pun 2009; Traff 1992; Banks and Toumazou 2008). The design reported by Tang and Pun (2009) is an inverter chain that uses a passive device such as a feedback loop to effectively boost the resolution to less than nano-ampere level. Since the resolution in our work only reaches to a micro-ampere level, considering design area and acceptable resolution, we use the inverter-based circuit design without passive device as the current comparator, shown in Figure 8(a). The input current is the current difference between the output current of LSTM and reference current through the interconnection. The inverter chain in the bottom is used for signal integrity and leaves enough time for the signal to reset. The inverter chain in the top feedback loop is to reset the voltage signal after a single comparison cycle. Setting the *W/L* of all n-channel MOSFETs and p-channel MOSFETs as 800nm/600nm and 1,200nm/200nm, respectively, Figure 8(b) shows the relationship between input current and delay. We observe maximum delay of 2.67ns when the current resolution is 1μA. For the design of current-mirror biasing the reference current used for the function of DAC in the back-end, we used p-MOSFET current mirror

Table 2. W/L Ratio of Transistors in the Circuits of Figures 6 and 7

Sigmoid		Hyperbolic tangent	
Transistor	W/L (nm/nm)	Transistor	W/L (nm/nm)
M_1	10,000/400	M_1	12,000/400
M_2	2,000/400	M_2	2,000/400
M_3	3,100/400	M_3	3,000/400
M_4	1,000/400	M_4	1,200/200
M_5	1,400/200	M_5	5,000/200
M_6	200/200	M_6	200/200
M_7	800/400	M_7	800/400
M_8	400/400	M_8	400/400
M_9	800/200	M_9	1,000/200
M_{10}	1,600/200	M_{10}	1,800/200
		M_{11}	400/400
		M_{12}	400/400



(a) (b)

Fig. 8. Current comparator used as back-end stage: (a) circuit diagram and (b) delay dependence on current amplification.

to match the signal polarity in our design. An inevitable problem is the decrease of current accuracy due to the channel modulation effect in MOSFETs (Allen and Holberg 2002). The reference current is the key signal to give correct digital signal output. Therefore, the channel lengths in all p-channel transistors of current mirrors are set to 800nm.

4.3 LSTM Operation

Figure 9 shows the system-level design of LSTM. We have divided the entire signal flow into four operations shown in Figure 9, which are described as follows: (a) Operation 1 uses front-end analog multiplier-based crossbars and interconnections to achieve DAC function and the term $\hat{\boldsymbol{W}}_{\boldsymbol{x}}\hat{\boldsymbol{x}}_{\boldsymbol{t}} + \hat{\boldsymbol{W}}_{\boldsymbol{h}}\hat{\boldsymbol{h}}_{\boldsymbol{t\text{-}1}}$; (b) In operation 2, results from the operation 1 will go through the blocks implementing activation functions or I/V converters to obtain all internal gate layers in the LSTM network; (c) Operation 3 uses analog multipliers without MIFG and the blocks of activation functions to obtain both $\boldsymbol{c}_{\boldsymbol{t}}$ and $\boldsymbol{h}_{\boldsymbol{t}}$; (d) Operation 4 uses current comparators with current mirrors to achieve ADC function. In this flow, analog multipliers, sigmoid blocks, and hyperbolic tangent blocks use 0.9V

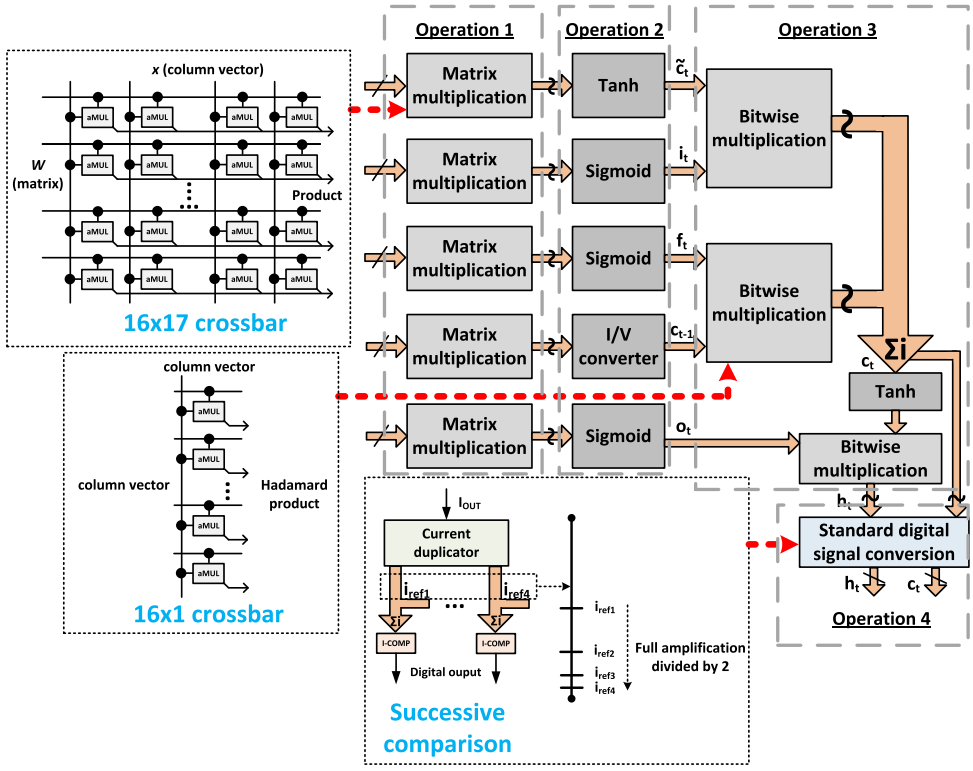


Fig. 9. System-level diagram of the proposed LSTM network.

and -0.9V as power supply and ground levels, respectively. For the current comparators used as back-end interface, they use 1.8V and 0V as power supply and ground levels, respectively.

The signal flow with slash arrows represents both input and output in standard digital domain. Internal signal flow with wavy lines represents current signals. The signal flow with sigma function mean the sum of current signals use the wire interconnection in physical implementation. For both matrix multiplication and bitwise multiplication to obtain Hadamard product, we use the proposed analog multiplier to build a crossbar, in which current results in each row are obtained by the wire interconnection. Especially, the matrix multiplication uses MIFG MOSFETs as the front-end to achieve DAC function. The rest of blocks do not use MIFG MOSFETs, since the signal flow has already been converted to an analog signal. This structure with only one stage connecting to MIFG not only avoids large loads for each analog block but also saves chip area. For the signal flow going through activation functions, it is current signal as the input and voltage signal as the output as described in the previous subsection. Note that for the term c_{t-1}, it directly comes from the previous cell without going through the matrix multiplication or an activation function. Thus, to obtain the correct signal flow, we still allow c_{t-1} to go through matrix multiplication with MIFG MOSFETs, which means c_{t-1} will be converted to an analog signal and multiplied by 1. Then current/voltage converter is applied to make c_{t-1} connect to the following crossbar of bitwise multiplication.

In our structure, the single analog current is converted to a 4-bit digital signal based on successive approximation. The method uses current signal to make a comparison with the reference current step-by-step to obtain multiple-bit as the final output. The proposed conversion flow is shown in Figure 9. First, we use the array of current mirrors as the current duplicator to generate

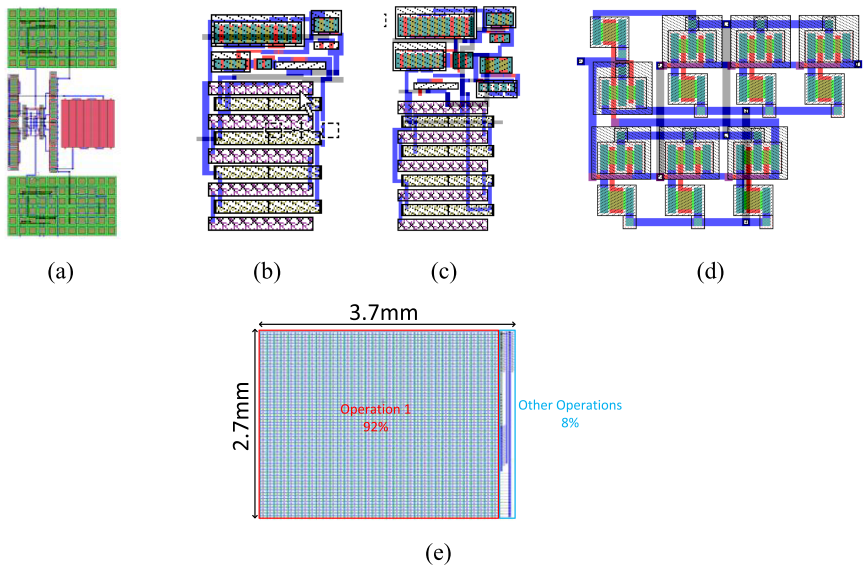


Fig. 10. Layout view: (a) proposed analog multiplier, (b) sigmoid function, (c) hyperbolic tangent, (d) current comparator, and (e) LSTM network.

four same-current signals representing $\boldsymbol{h_t}$. Then, using wire interconnection, we obtain the current difference between the reference current and the current signal of $\boldsymbol{h_t}$. The reference current in this step is set to 0μA, -90μA, -135μA, and -157.5μA, corresponding to MSB to LSB in our design. The first reference current can be simply obtained via connecting to the ground. The rest of current signals can be obtained through current mirrors. The last step is to use current comparators to compute the final 4-bit output. When the current difference is a positive value, the digital bit is high; otherwise, the digital bit is low. Compared to successive approximation register (SAR) ADC, we used an independent comparison block for each analog output to speed up data conversion instead of the single register with relative clock tree converting repeatedly. In addition, for the conversion of N-bit in flash ADC, it requires 2^{N-1} voltage comparators; while in our proposed conversion, it only requires N current comparators for each analog signal. In our design to implement an LSTM network, we only use MIFG MOSFETs in the first stage that executes matrix multiplication to convert external digital signals to analog signals, which are processed in internal blocks. In the internal computation, all addition behaviors are implemented by the simple physical interconnection of current signals to speed up computation. Once the data computation is finished, we use current comparators at low circuit cost instead of traditional methods as the back-end to achieve ADC function.

5 RESULT AND DISCUSSION

This section will evaluate the performance of the proposed LSTM network based on the described analog circuit design. The simulations and discussions cover the proposed computing core of the analog LSTM network and the core with memory access.

5.1 Analog LSTM Core

The physical design (layout) of all required sub-blocks in TSMC 180nm CMOS are shown in Figures 10(a)-(d). The entire design of the LSTM network with the matrix dimension of 16 × 16 is

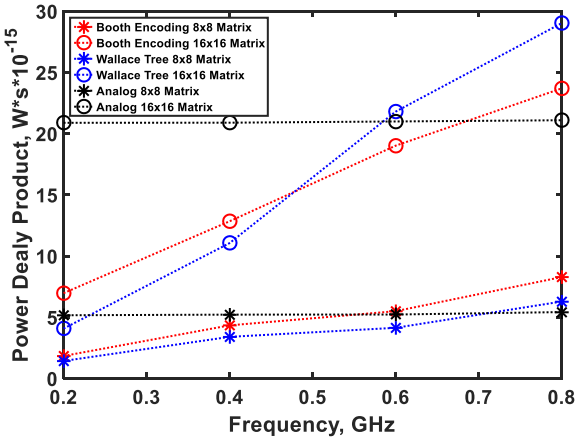


Fig. 11. PDP comparison between the proposed analog multiplier and other digital multipliers.

shown in Figure 10(e). The dimension of the chip layout is 3.7mm × 2.7mm. The crossbar used for both DAC function and matrix multiplication as the first stage contributes 92% of total chip area. This is due to the proposed crossbar that uses multiple inputs at the cost of area of capacitors and the use of dummy devices in the layout design of differential pairs and current mirrors in multipliers, which is also seen in Figure 10(a). The entire network is built by 16 independent computing paths to process analog signals of a 16 × 16 matrix in a parallel way. Thus, if some paths are set without input, then the dimension of the matrix to be processed will be reduced, in which the only power due to dc bias will be dissipated.

We have used analog multipliers instead of conventional digital multipliers to process the specific matrix multiplication, $N \times N$ matrix times N-column vector, which is the dominant calculation in computing of an LSTM network. It is necessary to compare two types of multipliers in terms of performance. We illustrate the comparison of power-delay product (PDP) of the matrix multiplication between the proposed analog multiplier and other digital multipliers, including the Booth encoding multiplier and Wallace tree multiplier (Fadavi 1992; Wallace 1964), as shown in Figure 11. We observe that when the operation frequency is low, both Booth-encoding and Wallace-tree multipliers show advantageous performance compared to the proposed one. With the work frequency boosting, the PDP of the proposed multiplier will be less than that of two digital multipliers. When the dimension is increased, the PDPs of two digital multipliers are much more sensitive to frequency boosting than that of the proposed analog design. This is due to the switch current, which is a strong function of frequency in high-speed digital circuits. For the proposed analog multiplier, it is always inputted by dc bias signal, and the only term varied by frequency is the small signal as the input signal to be computed. In the case of 16 × 16 matrix multiplication under 800MHz, the growth rate of PDP of matrix multiplication using Booth-encoding and Wallace-tree multipliers compared to that using the proposed one is 12.3% and 37.6%, respectively. From 200MHz to 800MHz, in cases of both 8 × 8 and 16 × 16 matrices, the growth rates of PDP of matrix multiplication using the proposed multiplier are all below 5%. In addition, the proposed design is capable of implementing DAC function as the front-end of the signal interface compared to previous work of analog multipliers. The proposed analog multiplier is suitable to work for large-dimension matrix multiplication under high frequency. No matter what the bit length of input is, the analog multiplier requires only one stage to obtain the product after the implementation of DAC function using multiple inputs. For the addition in matrix multiplication, the current carry signal going through the metal interconnection is used to speed up the addition without additional circuit block.

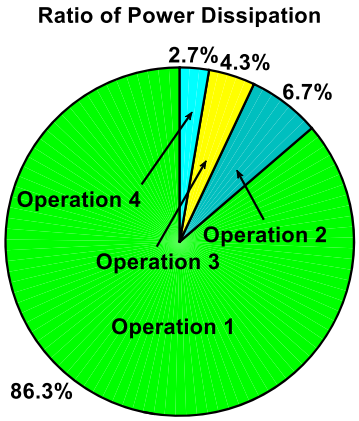


Fig. 12. Breakdown graph of power dissipation for each operation.

The single computation cycle in an analog LSTM network with no load is 0.87 to 1.19ns. The delay contributed by current comparators varies from 0.48 to 0.8ns, depending on computed current amplitude before the back-end. The rest of blocks contribute 0.39ns delay. Our design is a purely computing core with a feed-forward path and does not need an internal clock, on-chip memory, or internal sample and hold (S/H) circuit. Therefore, to avoid data competition, we restrict the maximum frequency of input signals to 800MHz. The total power dissipation of analog LSTM network with no load is 425.5mW. The four defined operations in Figures 9 and 12 show the breakdown graph of power dissipation for each operation. We see that the first operation to implement matrix multiplications consumes 86.3% of the total power dissipation, which is much larger than other operations. It is obvious, since matrix multiplication is the main calculation in an LSTM network. However, the proposed analog multiplier without feedback loop significantly reduces both power dissipation and latency, considering the design of signal interface. The computing flow built by several important transient simulations including internal and final results are shown in Figure 13. We see that the internal signals can be shifted fast as the input flipping continuously due to absence of S/H circuit. The final results of $\boldsymbol{c_t}$ and $\boldsymbol{h_t}$ including both logic high and logic low can be recognized even though there are some unwanted signal components presented.

5.2 Memory Access

The proposed LSTM design is purely a computing core and can only deal with memory-free computing of the 16×16 matrices. It is necessary to apply memory access to achieve block multiplexing extending the network depth. To address memory access, how to efficiently utilize the memory in terms of hardware resource, computing latency, and power dissipation is to be considered. For one weighted matrix in the proposed design, it has $(16 \times 17 \times 4)/8 = 136$ bytes. Therefore, for a single computing cycle, the memory requires $136 \times 8 = 1088$ bytes for all weighted matrices. Besides, for $\boldsymbol{c_t}$ and $\boldsymbol{h_t}$, these are required to be stored temporarily in memory for the usage of input in the next computing cycle, and $\hat{\boldsymbol{x}}_t$ updated in each computing cycle also uses memory for data hosting. Therefore, each of these three vectors requires the memory of $16 \times 4/8 = 8$ bytes. For the computing core of an LSTM network in one computing cycle, the maximum memory size is $1088+8 \times 3 = 1112$ bytes. For a trained LSTM network, if the pipelined topology is used for computing to extend the network depth, then these weighted matrices should be used for each computing cycle uniformly. $\boldsymbol{h_t}$ obtained from each computing cycle is not only used as the input of the next computing cycle, but it is also required to be stored for final prediction. If we define the network

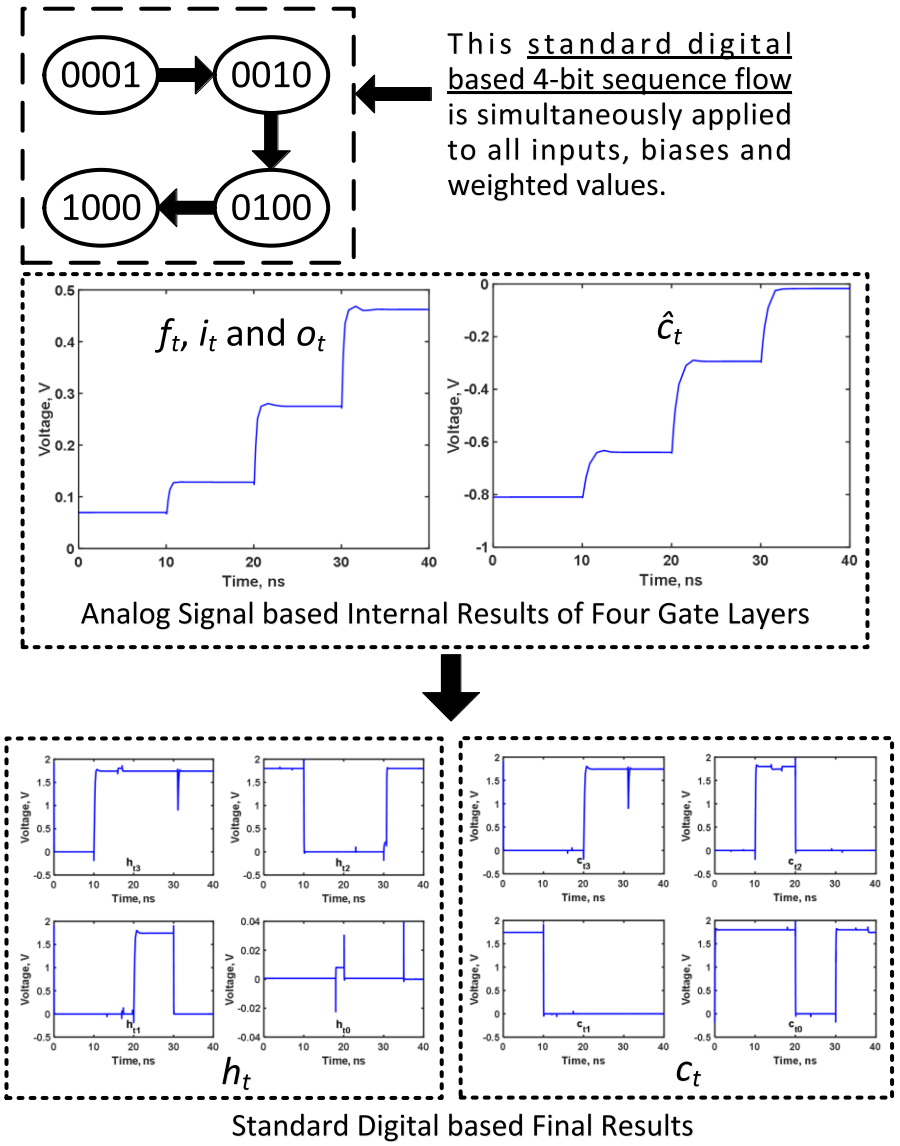


Fig. 13. Transient simulation based computing flow of LSTM network using the proposed ASP design.

depth, which is corresponding to the number of computing cycle as D_{LSTM}, based on the preceding description, then the weighted matrices after training occupy 1,088 bytes constantly. The memory size of $\hat{\boldsymbol{x}}_t$, $\boldsymbol{c}_t$, and $\boldsymbol{h}_t$ is 8 bytes, 8 bytes, and $(8 + 8D_{LSTM})$ bytes, respectively. Thus, the size of entire required memory is $(1112 + 8D_{LSTM})$ bytes. Since the utilized memory is small, the static random access memory (SRAM) in chip can be used considering overall latency. Figure 14 shows the overview of the proposed analog LSTM network with on-chip SRAM. It is seen that the SRAM block is the same as the normal one to operate write-in and read-out. The communication cycle between computing and memory is described as follows: (a) All weighted matrices, $\hat{\boldsymbol{x}}_t$, $\boldsymbol{c}_t$, and $\boldsymbol{h}_t$ are written into SRAM. $\boldsymbol{h}_t$ is written doubly for both computing in the next layer of the LSTM

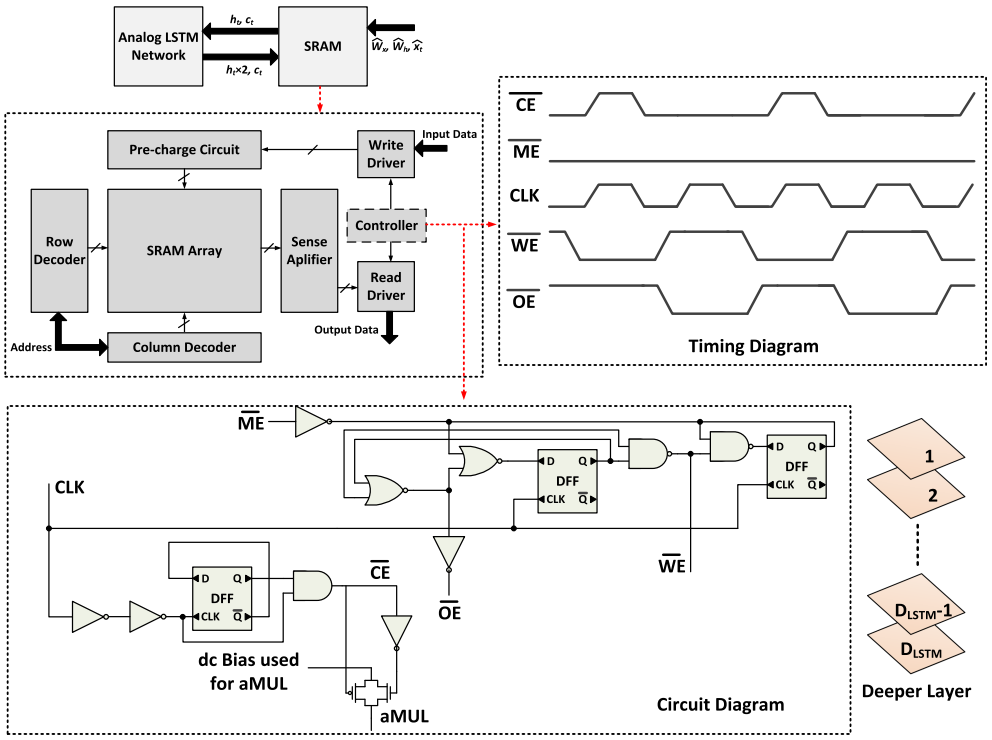


Fig. 14. Overview of the proposed analog LSTM network with on-chip SRAM access used for extending the depth of network.

network and final prediction. The analog LSTM computing core is idle during this write cycle; (b) All data stored in SRAM in the last cycle is read out to the analog LSTM core and the computing begins during this read cycle; (c) Above two clock cycles are seen as one computing cycle and repeated for obtaining a deeper LSTM network.

Using cycled flow represented in Figure 14, the network depth can be increased sequentially with SRAM access. Except $\boldsymbol{h_t}$ being stored for final prediction, the rest of useful data is stored only temporarily in SRAM for the computing of the next cycle, which reduces the workload in SRAM. To control the communication between the analog LSTM core and SRAM, the design of the memory controller should also be considered. The timing diagram of the controller is also shown in Figure 14, in which $\overline{OE}$, $\overline{WE}$, $\overline{CE}$, $\overline{ME}$, and CLK are enable ports of read-out, write-in, analog LSTM core and SRAM, and clock, respectively. In these enable ports, only $\overline{CE}$ is not the standard one. Figure 14 also shows the proposed circuit design of SRAM controller, in which $\overline{CE}$ is outputted by CLK. $\overline{OE}$ and $\overline{WE}$ are generated by CLK and $\overline{ME}$. Once $\overline{CE}$ is obtained, it can control the dc bias of the inputs of the proposed analog multipliers, which works for the analog LSTM network. During write-in, the analog LSTM core is idle and dc bias is disconnected through transmission gates controlled by $\overline{CE}$ to save energy. While SRAM is in read-out, dc bias is applied to the analog LSTM network via $\overline{CE}$ to let the computing work.

Using the same structure of SRAM as shown in Figure 14, in which 6T-cell is used for the SRAM cell, and the size of SRAM array is $128 \times 128 = 2$ kilobytes, as the test configuration of memory access. Figure 15 shows the simulation results of the analog LSTM network with SRAM access, depending on various network depths and frequencies. From Figure 15(a), we compare the power

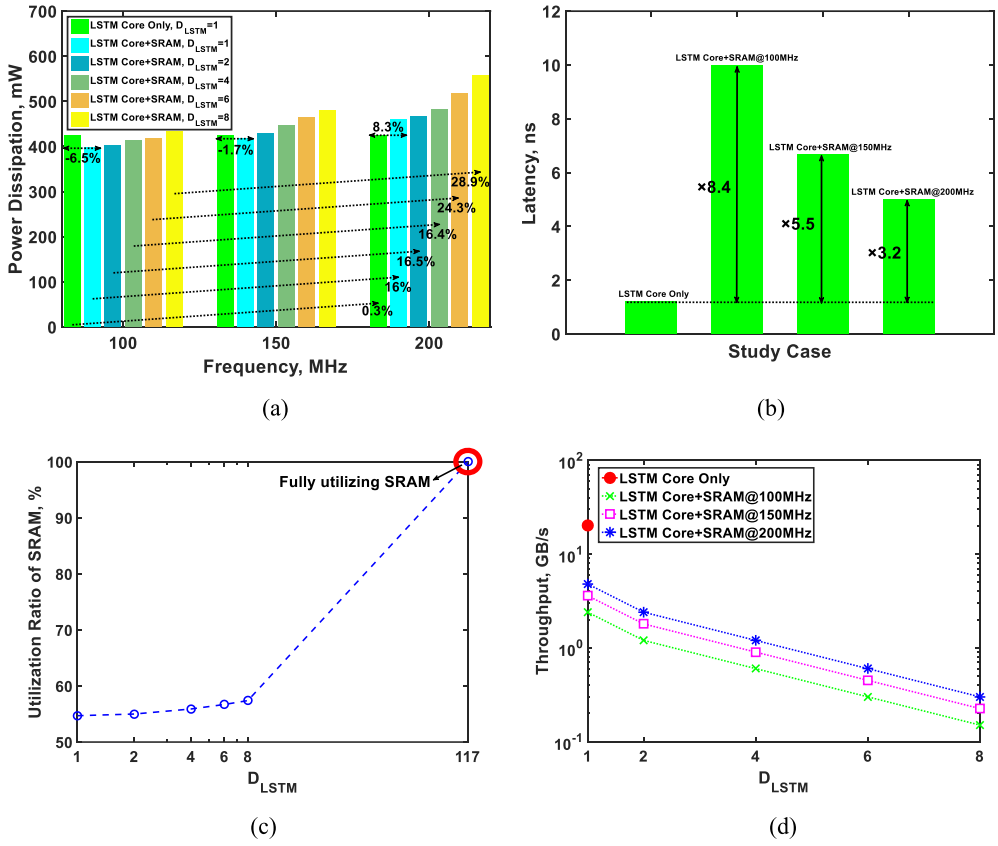


Fig. 15. Simulation results of the LSTM network with SRAM access: (a) power dissipation, (b) computing latency, (c) utilization ratio of SRAM, and (d) throughput.

dissipation when D_{LSTM} is 1. The overhead of power dissipation due to SRAM access is -6.5%, -1.7%, and 8.3% corresponding to clock frequency of 100MHz, 150MHz, and 200MHz, respectively. The reason that the overhead under 100MHz and 150MHz is negative during write-in is that the computing core is in idle state executed by the controller in SRAM and so only consumes static power. Therefore, from the view of average power dissipation given a full cycle of LSTM computing, the use of the analog LSTM network with SRAM can save power, compared to the analog LSTM network that works under non-clock without memory access. When the clock frequency increases to 200MHz, the SRAM access brings overhead of power dissipation to the proposed computing core due to the increase of dynamic power dissipation in SRAM. The overhead of power dissipation contributed by the proposed computing core is small. We can observe that from 100MHz to 200MHz, the increase of power dissipation in the proposed core is only 0.3% due to an increase in frequency of small signal. In the cases that the proposed core works with SRAM access, the growth rates of power dissipation due to frequency boosting are all increased. In the case that D_{LSTM} is 8, this growth rate is up to 28.9%. Figure 15(b) shows the overhead of latency due to SRAM access, in which the maximum overhead is × 8.4 under 100MHz. Actually, the introduced SRAM restricts the computing speed of the proposed analog LSTM core due to clock constraint and the access time of SRAM. As reported earlier, the peak latency of the proposed analog LSTM core without memory access is only 1.19ns, which means the maximum frequency can be up to

Table 3. Performance Comparison with Prior Work of LSTM

	Guan et al. 2017b	Ferreira and Fonseca 2016	Chang et al. 2017	Rybalkin et al. 2017	Conti et al. 2018	Han et al. 2016	Shin et al. 2017	Present work		
Platform	FPGA	FPGA	FPGA	FPGA	ASIC 65nm	ASIC 45nm	ASIC 65nm	ASIC 180nm		
Function	LSTM	LSTM	LSTM	LSTM	LSTM	CNN; LSTM	CNN; LSTM	LSTM		
Memory type	DDR3	SRAM	LUT; DDR3	LUT	SRAM	SRAM	LUT; REG	SRAM		
Clock frequency (MHz)	150	158.2	142	166	20	800	200	200		
Latency (ms)	390	4.6 × 10-4	-	2.37 × 103	-	7.5 × 10-3	-	10-5		
Bit length	32-bit float	18-bit fixed	16-bit fixed	5~16-bit float	8~16-bit fixed	4-bit fixed	4-bit fixed	4-bit fixed		
Depth of network	-	-	2	-	1	1	1	1	4	8
Peak power dissipation (mW)	19.63×10^3	420	2.3×10^3	104	1.24	590	21	460.3	481.6	557.4
Energy efficiency (TOP/s/W)	0.37×10^{-3}	-	1.46×10^{-4}	1.5×10^{-2}	3.08	0.17	1.1	0.95	0.9	0.78

800MHz. However, with the help of introduced SRAM, the LSTM core can be multiplexed to extend the depth of network. Since we have set the size of SRAM that is 2 kilobytes, the utilization ratio of SRAM should be studied. Figure 15(c) shows the results of these metrics. The utilization ratio of SRAM is from 54.6% to 57.4% corresponding to D_{LSTM} is from 1 to 8, respectively. Thus, the rest of unused SRAM cells suffers leakage current. Through the simulation of static power dissipation, this resource waste contributes 20 to 43mW as the variations of network depth and frequency. As predicted, when D_{LSTM} is 117, SRAM will be fully used. Another important issue is throughput between the proposed LSTM core and SRAM. Figure 15(d) shows these metrics. Here, we only consider the computed data for the computing in the next cycle, $\boldsymbol{h_t}$ and $\boldsymbol{c_t}$, and data stored in SRAM for the final prediction, $\boldsymbol{h_t}$. When the proposed core works uniquely, throughput is up to 20.1GB/s. Accessed with SRAM, the throughput is reduced due to clock constraint. In addition, the depth of network constrains the throughput. In the fastest operation at 200MHz, when D_{LSTM} is 8, the throughput is 0.3GB/s. However, under 100MHz, when D_{LSTM} is 8, the throughput is 0.15GB/s. The proposed LSTM core with on-chip SRAM can extend the depth of network at the cost of lowering computing speed. For the power issue, the proposed work sequence and its controller can let the LSTM core be idle during write-in state to save power. The disadvantage of the proposed design is low utilization ratio of SRAM, which also contributes to static power dissipation. But the size of SRAM in this work is only 2 kilobytes, and the utilization ratio of SRAM can be increased by increasing the depth of network.

Table 3 shows the performance comparison between our work and other reported work of LSTM based on digital implementation (FPGA and ASIC). The listed data of our work is from the computing core accessed with SRAM under 200MHz and included D_{LSTM} is 1, 4, and 8. Considering

the performance metrics of LSTM, quantization-bit, we observe that the LSTM implementation in FPGA platform is much more accurate than that in ASIC design, since the single computation block is used repeatedly at the cost of computation latency in FPGAs. Work in Guan et al. (2017b) uses off-chip memory to confirm accuracy when the matrix dimension becomes large. However, this strategy largely increases computation latency and power dissipation. Another FPGA-based work reported in Ferreira and Fonseca (2016) fully uses both logic units and memory cells in FPGA to speed up computation and suppress the power dissipation. Work in Chang and Culurciello (2017) balances the data communication that both on-chip LUT and off-chip DRAM are used for internal storage of matrix multiplication to reduce the latency due to off-chip memory access and workload of on-chip communication. Using digital signal to compute neural network, the longer bit length can cause larger power and latency. In the work of Rybalkin et al. (2017), the LSTM network with variable bit length is achieved to dynamically adjust both consumed energy and computing latency. Therefore, its peak energy efficiency is higher than other LSTM networks designed in FPGA. Summarizing this work based on FPGA, their energy efficiencies are all lower by at least 2 magnitudes than cited ASIC designs, but the bit length in these designs is very high compared to ASIC designs. For ASIC design, prior digital-based work (Shin et al. 2017; Han et al. 2016) and our presented analog-based work use a 4-bit fixed-point number to balance computing error and circuit cost. Both work of Han et al. (2016) and Shin et al. (2017) can compute both CNN and LSTM, which is more functional than our design. Han et al. (2016) uses weight encoding and matrix sparsity to reduce both power and required computing cycles when large matrices have to be processed. The error due to sparsity can be reduced by weight decoding and multi-index. Compared to Han et al. (2016), the boosting of energy efficiency of our design is 5.6 × corresponding to $D_{LSTM} = 1$. As mentioned in the introduction, the design in the work of Shin et al. (2017) mainly uses LUTs and registers to store all possible results of multiplication and activation functions, which removes complex real-time computing and its switching power. Thus, the energy efficiency of this work is higher than in our work. However, the final results must be obtained through multiple cycles. In our design, analog implementation without traditional data converters or feedback loop largely speeds up computation, which gives on-chip SRAM enough time for data access. For computing of a single-layer LSTM network (D_{LSTM}=1), it needs only one computing cycle in our work. A digital-based LSTM implementation called "Chipmunk" proposed in the work of Conti et al. (2018) shows higher bit length and energy efficiency than Shin et al. (2017) and our work. This is due to sparsity and matrix partitioning widely used in the design to largely reduce the unnecessary energy consumed during logic switching at the cost of higher error and more epochs of training to increase accuracy or perplexity. If our ASP-based design works for computing large matrices, then the strategy used in "Chipmunk" is a potential one if slight error can be tolerated. Considering latency, the proposed work has the least computing latency than any other LSTM work. Once again, our ASP-based work only uses one cycle to finalize one-layer computing, which is different from other work.

In Table 4, we also compare the proposed design with other work of ASP based on neural network (Bong et al. 2017; Amravati et al. 2018). In the work of Bong et al. (2017), ADC array is designed in another chip considering limited power budget in a single chip. In the computing chip, it uses multi-core to increase the computing efficiency. Since off-chip communication is introduced to increase the communication delay, the allowable clock frequency cannot be very high to avoid data race in the motherboard. Based on Table 4, the latency of this work is much higher than that of our work. Compared to the work of Bong et al. (2017), the boosting of energy efficiency of our design is 2 ×, corresponding to D_{LSTM}=1. Design in the work of Amravati et al. (2018) is suitable for applications in portable devices, and its energy efficiency is higher than in our work. In this work, digital-to-pulse converter (DPC) is used as signal interface. Thus, counter array has to be embedded for signal processing. Based on this principle, the final results cannot be obtained in a

Table 4. Performance Comparison with Prior Work of ASP Based on Neural Network

	Bong et al. 2017	Amravati et al. 2018	Present work		
Platform	ASIC 65nm	ASIC 55nm	ASIC 180nm		
Function	CNN	Autonomous control	LSTM		
Memory type	SRAM	-	SRAM		
Data conversion type	ADC	DPC	ADC; DAC		
Clock frequency (MHz)	100	20	200		
Latency (ms)	263	-	10^{-5}		
Power supply (V)	0.46~0.8	0.4~1	1.8		
Bit length	16-bit fixed	6-bit	4-bit fixed		
Depth of network	5	-	1	4	8
Peak power dissipation (mW)	211	0.69	460.3	481.6	557.4
Energy efficiency (TOP/s/W)	0.47	3.12	0.95	0.9	0.78

single computing cycle. Thus, the latency of this work cannot be lower than that of our design. Compared to this work, the novelty of our work is that we have proposed a signal interface instead of conventional ADC and DAC. For DAC, the only power consumed in the physical connection between a transistor's gate and MIFGs is due to resistances. For the back-end used as ADC, we use the array of current comparators to speed up the conversion and save chip area. Besides, the computing latency of the proposed LSTM core inside is fast without the restriction of clock or feedback loop, and we have used current signal to execute the row-addition for matrix multiplications instead of analog mixer.

5.3 Sensitivity Study

Another important issue is that analog implementation is more sensitive to the variations of process, voltage, and temperature (PVT) and noise than a digital implementation. The system used for the simulations of these non-ideal factors is the same as the one in the study of memory access. The work frequency is set at 200MHz. It is generally known that the real chip fabrication always has device mismatch. Thus, in the following sensitivity study, based on the Monte Carlo method, all simulations have a mismatch within ±20% distribution covering all transistors and passive devices. For the evaluation of PVT variation, the corner cases in this work include TTTT, SSHL, SFHL, FSHL, SSHH, SFHH, FSHH, and FFLH. Four letters from left to right in these cases represent the variations of n-type MOSFET, p-type MOSFET, temperature, and voltage supply, respectively. S and F located in the first two letters are slow and fast MOSFETs, respectively. T located in the first two letters represents typical MOSFETs following the original SPICE model provided by the foundry. We use the 3σ method to calculate the variable threshold voltage to define S and F for MOSFETs (McConaghy et al. 2013; Asenov 1998; Pelgrom et al. 1989). In this method, the standard deviation of threshold voltage can be represented by $\sigma_{th}=(qt_{ox}/\varepsilon_{ox}) \times (N_A W_d/4WL)^{1/2}$. H appearing in the last two letters is high temperature, 340K, or oversupply voltage, 1.85V. Accordingly, L is low temperature, 270K, or supply voltage with loss, 1.75V. T located in the last two letters represents default temperature in SPICE simulation, 300K, or standard supply power, 1.8V. We have introduced the fabrication mismatch and the corners representing variations of PVT. The last concern is noise introduced in our design. For CMOS technology, it includes two types of noise. The first one is device noise. For the analysis of a transistor or a resistor, it can be modeled using a paralleled current source. For the analysis of a capacitor, it can be modeled using a cascade voltage source. The detailed expressions of these noise sources have been deeply analyzed by Allen and Holberg

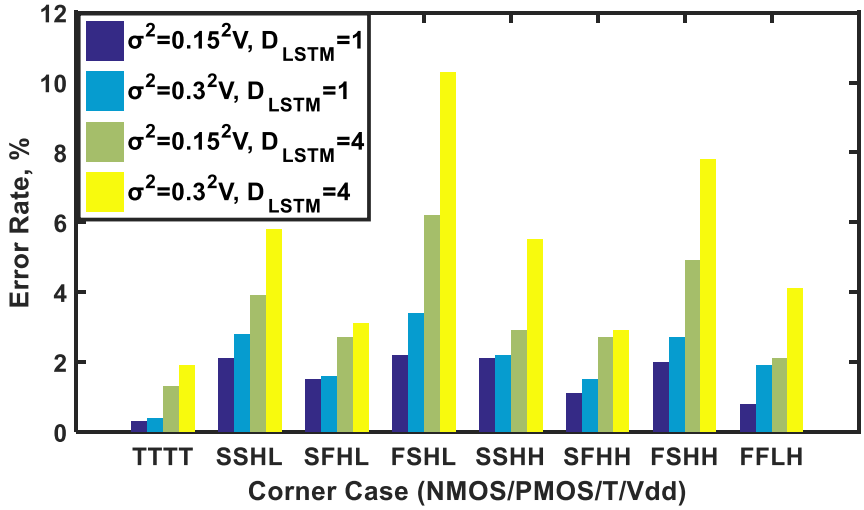


Fig. 16. Simulation results of error rate with PVT variation and introduced noise. All simulations are imported with mismatch of ±20% distribution of transistors, capacitors used for MIFGs and resistors.

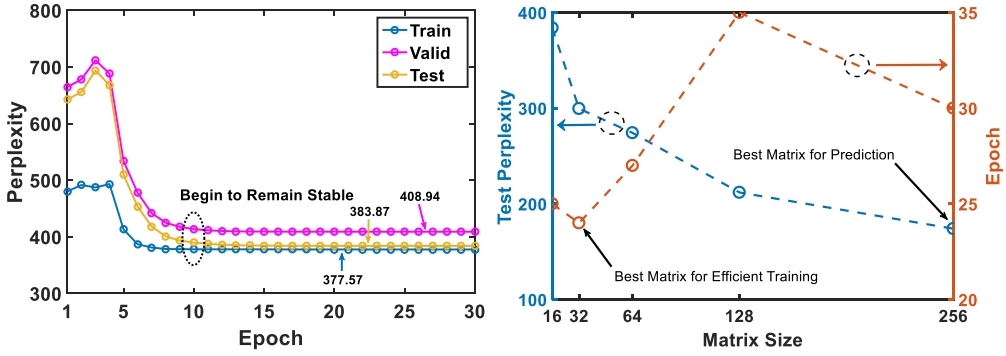


Fig. 17. Simulation results using the proposed LSTM structure with PTB dataset: (a) training result and (b) predictive results of perplexity and required epochs for stabilizing perplexity with matrix expanding.

(2002). The other one is the input noise that is imported from other blocks communicating with the proposed computing core; this one can be assumed to approximately follow Gaussian distribution, which is represented by $V_{in\_noise} = V_{in\_ideal} \times (1 + N(0,\sigma^2))$, according to the work of Long et al. (2018). We set σ^2 as 0.15^2V and 0.3^2V for the following simulations. Based on the configuration shown in Figure 17, we use error rate to evaluate the performance influenced by the described corner cases and noises. Figure 16 shows the simulation results. SSHL, FSHL, SSHH, and FSHH are four corners in which the error rate is higher than other corners. This observation follows that SS and FS for n-type and p-type transistors are the worst corners for digital VLSI. The performance of digital-based memory access in our work can be seriously influenced by these two corners. Thus, incorrect data communication can occur due to potential setup and hold violations. TTTT can be seen as an ideal case in our study, and so its error rate is the lowest. Considering digital blocks, FFLH is the fastest corner in our work. However, it shows a higher error rate than TTTT, which is due to the signal asynchronization between analog network without obvious acceleration and

the faster memory access. Considering the variation of supplied voltage, the error rate under over supplied voltage is less than that of undersupplied voltage with loss. Thus, it is necessary to avoid the loss of power supplying the proposed design. For $D_{LSTM} = 1$, the highest error rate is 3.4% corresponding to FSHL with the variance of input noise, $\sigma^2 = 0.3^2$V. For $D_{LSTM} = 4$, the highest error rate is 10.3%, also corresponding to FSHL with the variance of input noise, $\sigma^2 = 0.3^2$V.

5.4 Evaluation of Network Performance

We have used word-level language-prediction experiments to test the performance of the LSTM network using our ASP architecture. For an input sequence, the task is to predict the next word, given previous seen words. Our experiments are conducted on the Penn Tree Bank (PTB) dataset (Marcus et al. 1993), which consists of 929k training words, 73k validation words, and 82k test words. It has 10k words in its vocabulary. We stack our design to form a two-layer LSTM model and train it for 30 epochs using stochastic gradient descent. The hidden units of each LSTM layer, as our proposed architecture, is 16. Learning rate starts at 1.0, keeps steady for 4 epochs, and then decays 0.5 after each epoch. The training results are shown in Figure 17(a). From this figure, we can see that the perplexity keeps decreasing with epoch increases, and it almost remains stable after the 10th epoch. The best validation perplexity is at the 25th epoch. Train, valid, and test perplexities after being stable are 377.57, 408.94, and 383.87, respectively. The result shows that our LSTM architecture is capable of handling the tasks of real-world sequence prediction.

We also perform another predictive simulation experiment with different sized hidden units of the LSTM layer. The model is the same with the last experiment but only changes the number of hidden units. As Figure 17(b) shows, the test perplexity becomes smaller as the size of hidden units grows. When the matrix size is 256, the perplexity can be lowered to 174.06. The result verifies that the bigger size of hidden units in LSTM is necessary for improving the performance of the LSTM model. Considering the efficiency of training, we see that when the matrix size is 32, the proposed structure uses 24 epochs, which is smaller than any other case, to let the perplexity be stable. Mapping this prediction that larger matrices are used for computing, we need to expand the proposed ASP network. For $N \times N$ matrices used in the structure of our work, it requires N independent channels from front-end analog multipliers to back-end current comparators without additional power coupled by adjacent channels. Thus, the power overhead due to matrix expanding roughly shows a linearity to the number of channels. The computing latency will be mainly increased by current transportation in metals in the analog multiplier-based crossbar, which is very short compared to signal going through digital blocks. Considering memory access, using our ASP structure to connect with on-chip SRAM can increase the overhead of power and latency. Summarizing this matrix expanding, the increase of power is much larger than that of computing latency.

6 CONCLUSION

We have proposed a strategy to implement an LSTM network in analog circuit design. The entire analog-based computing core is compatible with external digital blocks. To speed up computing and reduce power dissipation, we have introduced a novel signal interface including MIFG MOSFETs and current comparators to achieve data conversion. In the computing core inside, we have used current-carry signal to reduce latency and remove the analog mixer used for row addition. The final post-layout simulation results show that the proposed analog LSTM network is suitable to work with on-chip standard memory. For the utilization of 2 kilobytes SRAM, the LSTM network can be computed to 117 layers. The finalization of one-layer computing requires only one computing cycle (two clock cycles). The overhead of power dissipation and computing latency due to memory access at 200MHz are 8.3% and 3.2×, respectively. The energy efficiency can be up to 0.95TOP/s/W. If more advanced CMOS technology is used for our design, then it is predicted

that both clock frequency and energy efficiency can be obviously increased. The sensitivity simulations considering noise, device mismatch, and PVT variation show that the worst case will lead to a 10.3% error rate in the proposed design corresponding to 4-layer LSTM computing. Overall, the proposed design shows acceptable performance amid these non-ideal factors. Using the same network structure of the proposed LSTM design with PTB as dataset for performance evaluation, the training requires 10 epochs to let all perplexities be stable. Besides, using the proposed LSTM structure with matrix expanding, lower perplexity under the simulation using PTB dataset can be obtained without large overhead of latency.

REFERENCES

P. E. Allen and D. R. Holberg. 2002. *CMOS Analog Circuit Design.* Oxford, New York.

K. M. Al-Ruwaihi. 1997. CMOS analogue neurone circuit with programmable activation functions utilizing MOS transistors with optimized process/device parameters. *IEE Proc. Circ., Devices and Systems* 144, 6 (1997), 318–322.

R. S. Amant, A. Yazdanbakhsh, J. Park, B. Thwaites, H. Esmaeilzadeh, A. Hassibi, L. Ceze, and Doug Burger. 2014. General-purpose code acceleration with limited-precision analog computation. In *Proceedings of the ACM/IEEE International Symposium on Computer Architecture (ISCA'14).* 505–516.

A. Amravati, S. B. Nasir, S. Thangadurai, I. Yoon, and A. Raychowdhury. 2018. A 55nm time-domain mixed-signal neuromorphic accelerator with stochastic synapses and embedded reinforcement learning for autonomous micro-robots. In *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC'18).* 124–126.

A. Asenov. 1998. Random dopant induced threshold voltage lowering and fluctuations in sub-0.1μm MOSFET's: A 3-D "atomistic" simulation study. *IEEE Trans. Electron Devices* 45, 12 (1998), 2505–2513.

A. Aslam-Siddiqi, W. Brockherde, and B. J. Hosticka. 1998. A 16×16 nonvolatile programmable analog vector-matrix multiplier. *IEEE J. of Solid-State Circ.* 33, 10 (1998), 1502–1509.

J. N. Babanezhad and G. C. Temes. 1985. A 20-V four-quadrant CMOS analog multiplier. *IEEE J. Solid-State Circ.* 20, 6 (1985), 1158–1168.

D. Banks and C. Toumazou. 2008. Low-power high-speed current comparator design. *Electronics Lett.* 44, 3 (2008), 171–172.

K. Bong, S. Choi, C. Kim, S. Kang, Y. Kim, and H. J. Yoo. 2017. A 0.62mW ultra-low-power convolutional-neural-network face-recognition processor and a CIS integrated with always-on haar-like face detector. In *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC'17).* 248–249.

A. X. M. Chang and E. Culurciello. 2017. Hardware accelerators for recurrent neural networks on FPGA. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS'17).* 1–4.

Y. H. Chen, T. Krishna, J. Emer, and V. Sze. 2016. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. In *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC'16).* 262–263.

F. Conti, L. Cavigelli, G. Paulin, I. Susmelj, and L. Benini. 2018. Chipmunk: A systolically scalable 0.9mm^2, 3.08Gop/s/mW @1.2mW accelerator for near-sensor recurrent neural network inference. In *Proceedings of the IEEE Custom Integrated Circuits Conference (CICC'18).* 1–4.

F. Corradi, C. Eliasmith, and G. Indiveri. 2014. Mapping arbitrary mathematical functions and dynamical systems to neuromorphic VLSI circuits for spike-based neural computation. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS'14).* 269–272.

J. Fadavi-Ardekani. 1992. M×N Booth encoded multiplier generator using optimized Wallace trees. In *Proceedings of the IEEE International Conference on Computer Design: VLSI in Computers & Processors.* 114–117.

J. C. Ferreira and J. Fonseca. 2016. An FPGA implementation of a long short-term memory neural network. In *Proceedings of the International Conference on Reconfigurable Computing and FPGAs (Recon Fig'16).* 1–8.

Y. Guan, H. Liang, N. Xu, W. Wang, S. Shi, X. Chen, G. Sun, W. Zhang, and J. Cong. 2017a. FP-DNN: An automated framework for mapping deep neural networks onto FPGAs with RTL-HLS hybrid templates. In *Proceedings of the IEEE International Symposium on Field-Programmable Custom Computing Machines (FCCM'17).* 152–159.

Y. Guan, Z. Yuan, G. Sun, and J. Cong. 2017b. FPGA-based accelerator for long short-term memory recurrent neural networks. In *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC'17).* 629–634.

G. Han and E. S. Sinencio. 1998. CMOS transconductance multipliers: a tutorial. *IEEE Trans. Circ. Syst. II* 45, 12 (1998), 1550–1563.

S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally. 2016. EIE: Efficient inference engine on compressed deep neural network. In *Proceedings of the ACM/IEEE Annual International Symposium on Computer Architecture (ISCA'16).* 243–254.

S. Hochreiter and J. Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780.

M. Holler, S. Tam, H. Castro, and R. Benson. 1989. An electrically trainable artificial neural network (ETANN) with 10240 "floating gate" synapses. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'89)*. 191–196.

C. W. Kim and S. B. Park. 1987. New four-quadrant CMOS analogue multiplier. *Electronics Lett.* 23, 24 (1987), 1268–1269.

Y. Long, T. Na, and S. Mukhopadhyay. 2018. ReRAM-based processing-in-memory architecture for recurrent neural network acceleration. *IEEE Trans. Very Large Scale Integr. Syst.* 99 (2018), 1–14.

D. Maliuk and Y. Makris. 2012. A dual-mode weight storage analog neural network platform for on-chip applications. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS'12)*. 2889–2892.

D. Maliuk and Y. Makris. 2015. An experimentation platform for on-chip integration of analog neural networks: a pathway to trusted and robust analog/RF ICs. *IEEE Trans. Neural Networks Learn. Syst.* 26, 8 (2015), 1721–1734.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Comput. Linguist.* 19, 2 (1993), 313–330.

T. McConaghy, K. Breen, J. Dyck, and A. Gupta. 2013. *Variation-Aware Design of Custom Integrated Circuits: a Hands-on Field Guide*. Springer Science+Business Media, New York.

C. Mead and I. Mohammed. 2012. *Analog VLSI Implementation of Neural Systems*. Springer Science & Business Media, New York.

B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst. 2017. Envision: A 0.26-to-10TOPS/W subword parallel dynamic voltage accuracy frequency scalable convolutional neural network processor in 28nm FDSOI. In *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC'17)*. 246–247.

L. Ni, Z. Liu, W. Song, J. Yang, H. Yu, K. Wang, and Y. Wang. 2017. An energy-efficient and high-throughput bitwise CNN on sneak-path-free digital ReRAM crossbar. In *Proceedings of the IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED'17)*. 1–6.

E. H. Nicollian and J. R. Brews. 1982. *MOS (Metal Oxide Semiconductor) Physics and Technology*. Wiley, New York.

B. A. Pearlmutter. 1989. Learning state space trajectories in recurrent neural networks. *Neural Computation* 1, 2 (1989), 263–269.

M. J. M. Pelgrom, A. C. J. Duinmaijer, and A. P. G. Welbers. 1989. Matching properties of MOS transistors. *IEEE J. Solid-State Circ.* 24, 5 (1989), 1433–1439.

F. Piazza, A. Uncini, and M. Zenobi. 1993. Neural networks with digital LUT activation functions. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'93)*. 1401–1404.

A. Pulver and S. Lyu. 2017. LSTM with working memory. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'17)*. 845–851.

J. Ramirez-Angulo, A. J. Lopez-Martin, R. G. Carvajal, and F. M. Chavero. 2004. Very low-voltage analog signal processing based on quasi-floating gate transistors. *IEEE J. Solid-State Circ.* 39, 3 (2004), 434–442.

V. Rybalkin, N. Wehn, M. R. Yousefi, and D. Stricker. 2017. Hardware architecture of bidirectional long short-term memory neural network for optical character recognition. In *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE'17)*. 1390–1395.

E. Sánchez-Sinencio. 2010. Floating Gate Techniques and Applications. Retrieved from http://ece.tamu.edu/~s-sanchez/607-2010-Floating%20Gate%20Circuits.pdf/.

A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar. 2016. ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars. In *Proceedings of the ACM/IEEE Annual International Symposium on Computer Architecture (ISCA'16)*. 14–26.

T. Shibata and T. Ohmi. 1992. A functional MOS transistor featuring gate-level weighted sum and threshold operations. *IEEE Trans. Electron Devices* 39, 6 (1992), 1444–1455.

R. L. Shimabukuro, M. E. Wood, and P. A. Shoemaker. 1991. A neural network synapse cell in 90nm SOS. In *Proceedings of the IEEE International SOI Conference*. 162–163.

D. Shin, J. Lee, J. Lee, and H. J. Yoo. 2017. DNPU: An 8.1TOPS/W reconfigurable CNN-RNN processor for general-purpose deep neural networks. In *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC'17)*. 240–241.

J. Sim, J. S. Park, M. Kim, D. Bae, Y. Choi, and L. S. Kim. 2016. A 1.42TOPS/W deep convolutional neural network recognition processor for intelligent IoE systems. In *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC'16)*. 264–265.

H. J. Song and C. K. Kim. 1990. An MOS four-quadrant analog multiplier using simple two-input squaring circuits with source followers. *IEEE J. Solid-State Circ.* 25, 3 (1990), 841–848.

D. C. Soo and R. G. Meyer. 1982. A four-quadrant NMOS analog multiplier. *IEEE J. Solid-State Circ.* 17, 6 (1982), 1174–1178.

A. Srivastava and C. Srinivasan. 2002. ALU design using reconfigurable CMOS logic. In *Proceedings of the IEEE Midwest Symposium on Circuits and Systems (MWSCAS'02)*. 663–666.

A. Srivastava and H. N. Venkata. 2003. Quaternary to binary bit conversion CMOS integrated circuit design using multiple-input floating gate MOSFETS. *Integration, the VLSI J.* 36, 3 (2003), 87–101.

T. C. Stewart and C. Eliasmith. 2014. Large-scale synthesis of functional spiking neural circuits. *Proc. IEEE* 102, 5 (2014), 881–898.

S. Subramanian. 2005. *Ternary Logic to Binary Bit Conversion Using Multiple Input Floating Gate MOSFETS in 0.5-Micron n-well CMOS Technology.* M.S.E.E. thesis. Louisiana State University.

S. Tabarce, V. G. Tavares, and P. G. de Oliveira. 2005. Programmable analogue VLSI implementation for asymmetric sigmoid neural activation function and its derivative. *Electronics Lett.* 41, 15 (2005), 863–864.

Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. 2014. DeepFace: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 1701–1708.

X. Tang and K. P. Pun. 2009. High-performance CMOS current comparator. *Electronics Lett.* 45, 20 (2009), 1007–1009.

H. Traff. 1992. Novel approach to high speed CMOS current comparators. *Electronics Lett.* 28, 3 (1992), 310–312.

H. N. Venkata. 2002. *Ternary and Quaternary Logic to Binary Bit Conversion CMOS Integrated Circuit Design using Multiple Input Floating Gate MOSFETs.* M.S.E.E. thesis, Louisiana State University.

C. S. Wallace. 1964. A suggestion for a fast multiplier. *IEEE Trans. Electronic Comput.* 13, 1 (1964), 14–17.

R. Wang, C. S. Thakur, T. J. Hamilton, J. Tapson, and A. van Schaik. 2016. A stochastic approach to STDP. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS'16).* 2082–2085.

R. Wang, T. J. Hamilton, J. Tapson, and A. van Schaik. 2014. A compact neural core for digital implementation of the neural engineering framework. In *Proceedings of the IEEE Conference on Biomedical Circuits and Systems (BioCAS'14).* 548–551.

Received April 2018; revised August 2018; accepted October 2018