Calibration Method to Reduce the Error in Logarithmic Conversion with Its Circuit Implementation

Zhou Zhao [1], Ashok Srivastava [1*], Lu Peng [1], Saraju P. Mohanty[2]

[1] Division of Electrical and Computer Engineering, Louisiana State University, Baton Rouge, U.S.A.
[2] Department of Computer Science and Engineering, University of North Texas, Denton, U.S.A.
[*] eesriv@lsu.edu

Abstract: In this paper, based on Mitchell's logarithmic conversion, we propose a fast calibration method using a fixed binary code with case judgement, which suppresses the conversion error. We developed a highly paralleled circuit serving the proposed calibration method. Differential cascade voltage switch logic (DCVSL) is used to work in both high-speed logic and adiabatic logic and trade-off between power dissipation and operation speed. In addition, a low cost adiabatic clock generator without any passive component is presented to support a four-phase sine clock for the adiabatic logic operation. An 8-bit logarithmic converter is designed in TSMC 180nm CMOS. Simulation results show that the proposed calibration can reduce the conversion error to 1.55% based on Mitchell's algorithm, the power dissipation varies between 1.12-3.709mW and the delay is 1.82ns under operational DCVLS.

1. Introduction

Multi-bit logic appears everywhere in electrical systems, most of which require large circuit area and long computation time. In some specific applications, system can tolerate a little bit error that does not influence the performance visibly [1, 2]. Thus, some approximated computations can be applied in specific applications. A very interesting theory is that a binary number can be converted to the form of a logarithmic code which can be processed by the following task with conversion error occurred. This theory has been used to simplify multiplication and division [3, 4]. [Reply to Reviewer-1, comment-2] A lot of work has been done to reduce the conversion error. One strategy is to use a multi-step function to further increase the conversion accuracy [5-8]. SanGregory, *et al.* [9] used another curve to approximate the logarithmic curve instead of the linear function used for the generation of the fraction part. Iteration can mitigate the approximated error step-by-step at the cost of long computation time [10]. Lookup table (LUT) is also used to supress the conversion error [11] but it introduces additional memory array. In addition, the search method for the logarithmic conversion has been studied with the purpose of effectively using LUT [12]. Programmable logic array (PLA) has also been used [13, 14]. This method can accelerate the design but it does not introduce the circuit optimization due to the restriction of already-defined logic gates in PLAs. For the VLSI circuit design, the leading-one detector is widely used for determining both the integer part and fraction part [15, 16, 17]. The leading-one detector, (Fig. 1 in [15]) is the pipelined circuit using MUX and AND gates to search the first high-bit for any given binary number. The computation time is highly dependent on both the word length and the location of the first high-bit.

The power dissipation is another important issue in the circuit implementation. For energy efficient VLSI design, adiabatic logic is one of promising logic families that is well used [18, 19]. The essence of the adiabatic logic is to activate slow charge/discharge under clocked power supply to reduce the power dissipation [20]. In this work, we introduce a fixed-binary method to calibrate the conversion error based on Mitchell's logarithmic conversion [3] and present a highly paralleled circuit structure of logarithmic converter which can be supplied by both dc power and clocked power to verify the proposed calibration method.

The main work is summarized as follows:

i. We analysed the classical Mitchell's algorithm used for logarithmic conversion. Based on this, we proposed a novel calibration using a fixed binary code with a case judgement to reduce the conversion error.

ii. We present a highly paralleled circuit structure for fast logarithmic converter with the proposed calibration method. We used differential cascade voltage switch logic (DCVSL) as the bottom gate. The proposed topology can be powered either by dc voltage for fast computation or by the clocked power to perform as an efficient charge recovery logic (ECRL) at the cost of lowering the work speed. For calibration block, we used a specific digital comparator, which can be used only to compare with a fixed code to implement the case judgement of the proposed calibration, and a simplified carry lookahead adder (CLA) to speed up the calibration. To reliably drive the converter, we proposed a ring oscillator-based circuit without any passive device to generated four-phase sine wave for the adiabatic logic.

iii. We compared the proposed calibration and other algorithms of logarithmic conversion. We designed an 8-bit logarithmic converter using TSMC 180nm technology. Post layout simulation shows that the proposed calibration method is robust and supported by our circuit with a wide working range. The power dissipation of the entire logarithmic converter under adiabatic logic is much smaller than that under DCVSL.

In Section 2, we present the analysis of the Mitchell's logarithmic conversion and propose a novel calibration method to supress the conversion error. In Section 3, we discuss the circuit structure to serve the logarithmic converter with the consideration of both the proposed calibration method and the purpose of speeding up adiabatic logic.

Section 4, we discuss simulation results and comparison with the prior work. The conclusions are presented in Section 5.

2. Study of calibration for logarithmic conversion

2.1. Mitchell's logarithmic conversion

The logarithmic multiplication and division are described below,

$$\log_2 f_{mul} = \log_2(a \times b) = \log_2 a + \log_2 b \quad (1)$$

$$\log_2 f_{div} = \log_2(a \div b) = \log_2 a - \log_2 b \quad (2)$$

Above two equations show that multiplication and division in binary domain can be converted to addition and subtraction in logarithmic domain, respectively. This logarithmic conversion can reduce the design complexity and the computation latency in the circuit implementation, which is a significant motivation of processing some specific operations in logarithmic domain.

For any number, it can be factored as follows,

$$X = 2^{X_I}(1 + X_F) \quad (3)$$

Taking logarithmic conversion of (3), we obtain:

$$\log_2 X = X_I + \log_2(1 + X_F) \quad (4)$$

where X_I is the integer part of the converted number and $\log_2(1+X_F)$ contributes the fraction part of the logarithm code. When X_F is between 0 and 1, $\log_2(1+X_F)$ can be approximated to X_F. Thus, the converted number can be shown as follows,

$$\log_2 X \approx X_I + X_F \quad (5)$$

This is the fundamental theory of Mitchell's logarithmic conversion [3]. The conversion process can be summarized as follows:

1) Convert the decimal number into a binary number and search for the MSB.

2) Use the digit of the MSB as the integer part of logarithmic code.

3) Copy all of numbers after MSB as the fraction part of logarithmic code.

As an example, take the decimal number 21. Its binary form is 10101. The MSB is in the fourth digital position. According to the described principle, the integer part is 100, and the bits after MSB, 0101 is the fraction part in logarithmic code. Therefore, the converted number is $(100.0101)_2$. It should be noticed that the converted numbers have an error, which will affect the accuracy of the following multiplication or division as analysed in [21]. The conversion error for a single number can be shown as follows,

$$\varepsilon = \log_2(1 + X_F) - X_F \quad (6)$$

If there is no calibration to reduce above conversion error, the average conversion error for a single binary number is 0.0572.

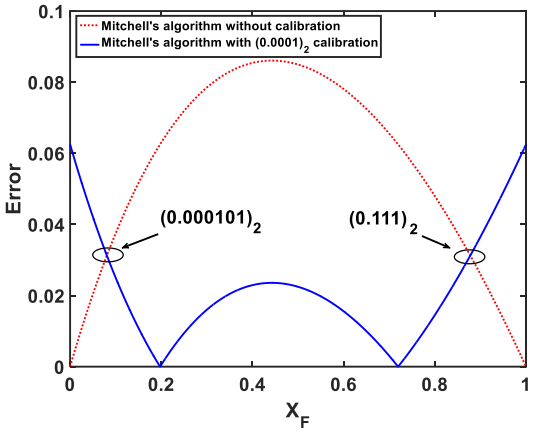
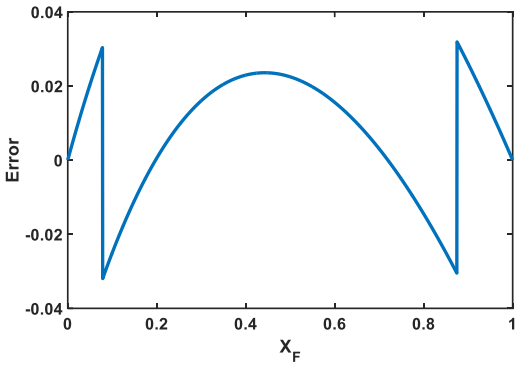
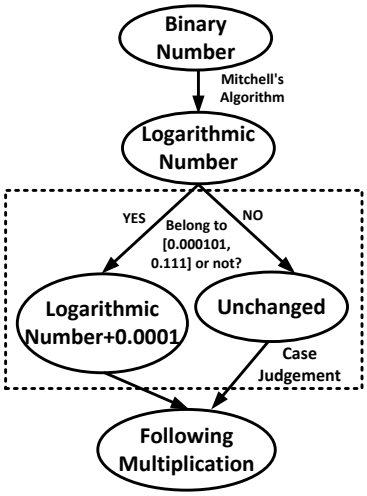


Fig. 1. *Error curves of Mitchell's algorithm and the one with the addition of C*



a



b

Fig. 2. *Proposed calibration with case judgement*
a Error curve
b Calibration flow

2.2. The proposed calibration

From (6), if the base in logarithmic form is 2, the error is always positive and occurs between $2k$ and $2k+1$, where k is any positive integer.

In (6), if we add a positive binary code to X_F, thereby shifting the error curve down, the conversion error can be described as follows,

$$\frac{\partial \varepsilon_{cal\_avg}}{\partial C} = \frac{\dfrac{-2^{C-1}(\ln 2)^2}{-2^{C-1}\ln 2 + e^{W\left(-2^{C-1}\ln 2\right)}}}{\ln 2}\left\{-2-4C-\frac{4}{\ln 2}+\frac{2}{\ln 2}\ln\left[3+\frac{W\left(-2^{C-1}\ln 2\right)}{\ln 2}\right]\right. \tag{11}$$
$$\left.+\frac{2}{\ln 2}\ln\left[-\frac{W\left(-2^{C-1}\ln 2\right)}{\ln 2}\right]+\left[6+\frac{2}{\ln 2}W\left(-2^{C-1}\ln 2\right)\right]\left[\frac{1}{3\ln 2+W\left(-2^{C-1}\ln 2\right)}\right]+\frac{2}{\ln 2}\right\}-\frac{4W\left(-2^{C-1}\ln 2\right)+5\ln 2}{\ln 2}$$

$$\varepsilon_{cal} = \log_2(1+X_F)-(X_F+C) \tag{7}$$

where ε_{cal} is the new error and C is the calibration code. The average conversion can be computed as follows,

$$\varepsilon_{cal\_avg} = \frac{1}{(X_{high}-X_{low})}\int_{X_{low}}^{X_{high}}\left[\log_2(1+X_F)-(X_F+C)\right]dX_F \tag{8}$$

Eqn. (8) indicates that X_F is located in the interval [X_{low}, X_{high}]. Since the integral is applied to a pure fraction number, X_{low} and X_{high} are 0 and 1, respectively. Note that we choose an absolute value for the integral and in that the introduced binary number will make some part of the conversion error to be negative. If only original value is integrated, the calculated average value is not the real error. Absolute deviation only can reflect the real error.

Setting (7) to 0, we obtain a transcendental equation. Using Newton-Raphson method to find the root step by step, two roots can be obtained which are:

$$\begin{cases} X_{F1} \approx \dfrac{-W\left(-2^{C-1}\ln 2\right)}{\ln 2}-1 \\ X_{F2} \approx \dfrac{W\left(-2^{C-1}\ln 2\right)}{\ln 2}+2 \end{cases} \tag{9}$$

where W(x) is the product logarithm function. To make both roots to be the real numbers, the variable in the product logarithm function, $-2^{C-1}\ln 2$, should be larger than -1/e [22]. Under this restriction, C should be smaller than 0.0861. Eqn. (9) gives two subsection points of the new error curve crossing the X-axis. Thus, (8) can be rewritten as follows,

$$\begin{aligned}\varepsilon_{cal\_avg} &= \int_0^{\frac{-W(-2^{C-1}\ln 2)}{\ln 2}-1}\left[X_F+C-\log_2(1+X_F)\right]dX_F \\ &+\int_{\frac{-W(-2^{C-1}\ln 2)}{\ln 2}-1}^{\frac{W(-2^{C-1}\ln 2)}{\ln 2}+2}\left[\log_2(1+X_F)-(X_F+C)\right]dX_F \\ &+\int_{\frac{W(-2^{C-1}\ln 2)}{\ln 2}+2}^{1}\left[X_F+C-\log_2(1+X_F)\right]dX_F \\ &= \frac{W\left(-2^{C-1}\ln 2\right)\left(-2-4C-\frac{4}{\ln 2}\right)-(5\ln 2)C-4.5\ln 2-5}{\ln 2} \\ &+\frac{\left[6+\frac{2}{\ln 2}W\left(-2^{C-1}\ln 2\right)\right]\ln\left[3+\frac{W\left(-2^{C-1}\ln 2\right)}{\ln 2}\right]}{\ln 2} \\ &+\frac{\left[\frac{2}{\ln 2}W\left(-2^{C-1}\ln 2\right)\right]\ln\left[-\frac{W\left(-2^{C-1}\ln 2\right)}{\ln 2}\right]}{\ln 2}\end{aligned} \tag{10}$$

Taking derivative of above equation with respect of C, we obtain the equation shown in (11).

For the variable C, in the interval [0, 0.063], (11) is negative, while in the interval [0.063, 0.0861], it is positive, so the minimum average error is around 0.0063. Converting this decimal number to a binary number, $(0.00010000001)_2$, is the highly approximated option. Considering that the longer bits contributing to the calibration lead to a more complex circuit block with higher power dissipation, and the LSB in $(0.00010000001)_2$ is a very small fraction number, we remove the LSB and set C to $(0.0001)_2$ as the fixed binary code to implement the calibration.

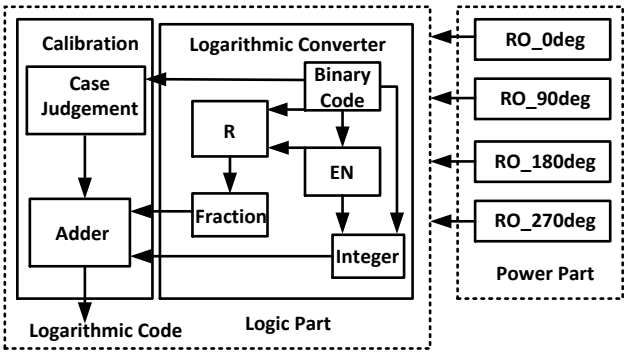


Fig. 3. *Block diagram of the proposed logarithmic converter*

Fig. 1 shows error curves of Mitchell's algorithm and the one, which is based on Mitchell's algorithm with the addition of C. We notice that when the fraction part, X_F approaches to 0 or 1, the error with $(0.0001)_2$ calibration is larger than that without calibration. Thus, there should be no calibration for X_F in these two areas, which are marked by two subsection points of two curves, $(0.000101)_2$ and $(0.111)_2$, both of which are used to build a piecewise function for the final calibration. Fig. 2 shows this error curve and its calibration flow. The proposed calibration only uses a fixed calibration code with a simple case judgement to supress the conversion error. When X_F is between $(0.000101)_2$ and $(0.111)_2$, the calibration is enabled. Otherwise, the traditional conversion based on Mitchell's algorithm is still applied without the proposed calibration. Under this piecewise function, the average error computed by Matlab is 0.0155.

3. Circuit implementation

In this section, we introduce a highly paralleled circuit structure for the logarithmic converter with the proposed calibration as shown in Fig. 3. The entire system mainly consists of two parts: logic part and clock part. Logic part works for the logarithmic converter designed by DCVSL. It uses two bridge signals named as *EN* and *R*, both of which will be introduced later, to obtain non-calibration logarithmic code. The calibration uses case judgement and adder to obtain the final logarithmic code. The power part is designed to supply dc power or four-phase clocked power.

3.1. Adiabatic logic

The way of designing digital logic using DCVSL under dc bias is same as using static logic gates [23]. When

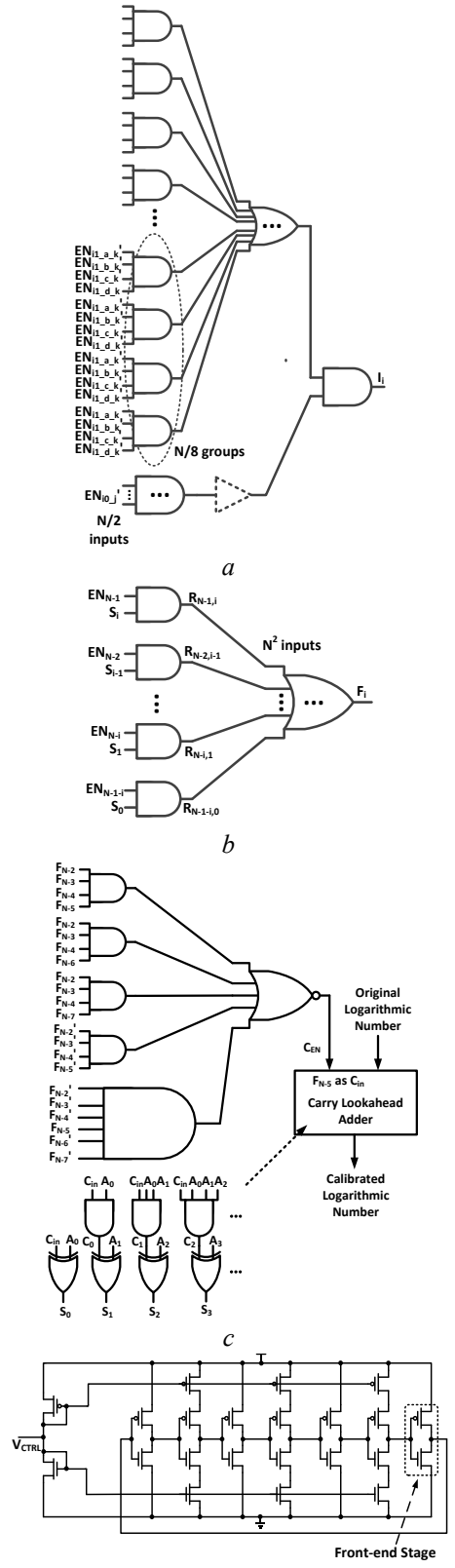


Fig. 4. *Circuit diagram of the proposed N-bit logarithmic converter*
a Integer part
b Fraction part
c Calibration block
d Clock generator

supplying clocked power to the whole system, DCVSL is changed to ECRL behaving as an adiabatic logic [24].

Adiabatic logic has been introduced in VLSI design considering the power budget and the limitation of cooling technique. Under clocked power supply, the single work cycle in ECRL is built by four phases, which are evaluation phase, hold phase, recover phase and wait phase. The evaluation phase and recover phase correspond to slow charge and discharge, respectively. For the two connected logic stages, the second stage must work under the next phase compared to the first stage. Therefore, to ensure that the entire logic chain processes correct data transmission, both phases of slow charge and discharge should work one by one in cascade way. The highly pipelined clocked power must supply to the entire cascade chain with 90° phase difference. It means an adiabatic system needs to add dummy buffers to build a strict pipelined logic chain to confirm each path has the same number of logic stages from the initial node to the terminal node.

Another issue is regarding the highest fan-in for a single gate. We follow a paralleled structure to supress computation delay. Thus, the drive-in ability should be seriously taken into consideration. Using a high fan-in structure, it can supress the delay at the cost of signal integrity [25]. Making a trade-off between signal integrity and delay, we restrict the upper bound of fan-in to eight for a single gate.

3.2. Logic part

The logic part is for the logarithmic conversion with a calibration block. The logarithmic converter can be split into two parts: integer conversion and fraction conversion. For the integer conversion, the leading-one detection and overflow detection [15, 16, 17] are widely used to find the MSB of a given binary number. Both of two strategies require a long logic chain with register, which brings long time to get the results and is sensitive to the external clock signal. Thus, it is required to find another way to achieve a fast integer conversion.

For a given N-bit binary number, $S_{N-1}S_{N-2}\ldots S_1S_0$, if MSB is located in S_{MSB} bit, higher bits than S_{MSB} are all 0. This means,

$$\overline{S}_{N-1} \bullet \ldots \overline{S_{MSB+1}} \bullet S_{MSB} = 1 \tag{12}$$

using (12), we create a series of enable signals to help both integer conversion and fraction conversion. The enable signal, EN_i, is expressed as follows,

$$EN_i = \begin{cases} \overline{S_{N-1} + S_{N-2} + \ldots S_1} & MSB = S_0 \\ \overline{S_{N-1} + \ldots S_{MSB+1}} + \overline{S_{MSB}} & MSB \in [S_1, S_{N-2}] \\ S_{N-1} & MSB = S_{N-1} \end{cases} \tag{13}$$

Eqn. (13) indicates that for random numbers with a fixed MSB, the introduced enable signals are always unique that one is high logic and the rest of them are low logic. Thus, for a group of numbers with the same MSB, their integer parts in logarithmic codes are the same. For the circuit implementation, the enable signals can be obtained through NOR array guided by (13). We can use enable signals to obtain the corresponding integer number as follows,

$$I_i = (\prod_{j=1}^{\frac{N}{2}} \overline{EN_{i0\_j}}) \bullet \sum_{k=1}^{\frac{N}{8}} \begin{pmatrix} \overline{EN_{i1\_a\_k}} \bullet \overline{EN_{i1\_b\_k}} \bullet \overline{EN_{i1\_c\_k}} \bullet EN_{i1\_d\_k} \\ + \overline{EN_{i1\_a\_k}} \bullet \overline{EN_{i1\_b\_k}} \bullet EN_{i1\_c\_k} \bullet \overline{EN_{i1\_d\_k}} \\ + \overline{EN_{i1\_a\_k}} \bullet EN_{i1\_b\_k} \bullet \overline{EN_{i1\_c\_k}} \bullet \overline{EN_{i1\_d\_k}} \\ + EN_{i1\_a\_k} \bullet \overline{EN_{i1\_b\_k}} \bullet \overline{EN_{i1\_c\_k}} \bullet \overline{EN_{i1\_d\_k}} \end{pmatrix} \quad (14)$$

In (14), I_i reflects a single bit in the integer part of a logarithmic code. If an N-bit binary number follows $N=2^{n+1}-1$, I_i ranges from I_0 to I_n. EN_{i0_j} is the group of enable signals that map I_i to be low logic in truth table. In this group, there are $N/2$ enable signals for any I_i to build the first term in (14). $EN_{i1\_(a, b, c, d)\_k}$ is the group of enable signals that map I_i to be high logic in truth table. In this group, there are also $N/2$ enable signals for any I_i. Every 4 $EN_{i1\_(a, b, c, d)\_k}$ signals in Sigma function in (14) are grouped again, that begins with the first $EN_{i1\_a\_k}$ appeared and then sorted one by one in truth table, to form the second term of in (14). Using the binary code, 00101101, as the case study of (14). Only EN_5 corresponds to high logic and the rest of EN_i to low logic. From the truth table of EN_i and mapping to I_i and using (14), we can determine the groups of both EN_{i0} and EN_{i1} as follows,

$$I_2=(E_7 \cdot E_6' \cdot E_5' \cdot E_4' + E_7' \cdot E_6 \cdot E_5' \cdot E_4' + E_7' \cdot E_6' \cdot E_5 \cdot E_4' + E_7' \cdot E_6' \cdot E_5' \cdot E_4) \cdot (E_3' \cdot E_2' \cdot E_1' \cdot E_0'), \quad (15a)$$

$$I_1=(E_7 \cdot E_6' \cdot E_3' \cdot E_2' + E_7' \cdot E_6 \cdot E_3' \cdot E_2' + E_7' \cdot E_6' \cdot E_3 \cdot E_2' + E_7' \cdot E_6' \cdot E_3' \cdot E_2) \cdot (E_5' \cdot E_4' \cdot E_1' \cdot E_0'), \quad (15b)$$

$$I_0=(E_7 \cdot E_5' \cdot E_3' \cdot E_1' + E_7' \cdot E_5 \cdot E_3' \cdot E_1' + E_7' \cdot E_5' \cdot E_3 \cdot E_1' + E_7' \cdot E_5' \cdot E_3' \cdot E_1) \cdot (E_6' \cdot E_4' \cdot E_2' \cdot E_0'). \quad (15c)$$

Using above three specific equations with all EN_i of the given number, the integer part can be obtained, which is 101 as desired. The circuit implementation of (14) is shown in Fig. 4a. The logic gates with three dotted symbols indicate that the single gate may extent to the logic chain if fan-in exceeds 8. The buffer shown by the dotted symbol is a dummy buffer or a chain of buffers, which makes the entire block to correctly work under ECRL.

To get the fraction part, we also try to find a paralleled topology. If we define an intermediate term $R_{m,n}$, which can be expressed by,

$$R_{m,n} = EN_m \bullet S_n \quad (16)$$

Using this intermediate signal, we can write,

$$F_i = \sum_{j=0}^{i} R_{N-1-j,i-j} \quad (17)$$

In (17), F_i refers to any single bit in the fraction part of a logarithmic code, which is implemented by the circuit shown in Fig. 4b. The final form of the logarithm code converted from the N-bit binary code can be expressed as $I_nI_{n-1}\ldots I_1I_0.\ F_{N-2}F_{N-3}\ldots F_1F_0$.

The calibration block is to reduce the error due to Mitchell's logarithmic conversion. As described in previous section, whether the proposed calibration is enabled or not, is strictly based on the original converted code. Thus, we need to compare the converted code with the lower bound and upper bound of calibration region. Digital comparator is widely used for the binary code comparison [26]. Our calibration requires the fraction part to be compared only with the fixed reference numbers $(0.000101)_2$ and $(0.111)_2$. Thus, we give up the standard digital comparator but use a simple logic, which can compare with our fixed bounds at low circuit cost.

For a single converted number, first a signal, C_{EN} is defined, to judge if the number locates between upper bound and lower bound in which the number needs to be calibrated. For a given N-bit input, C_{EN} can be expressed as follows:

$$C_{EN} = \overline{\begin{matrix} F_{N-2} \bullet F_{N-3} \bullet F_{N-4} \bullet F_{N-5} + F_{N-2} \bullet F_{N-3} \bullet F_{N-4} \bullet F_{N-6} \\ + F_{N-2} \bullet F_{N-3} \bullet F_{N-4} \bullet F_{N-7} + \overline{F_{N-2}} \bullet \overline{F_{N-3}} \bullet \overline{F_{N-4}} \bullet \overline{F_{N-5}} \\ + \overline{F_{N-2}} \bullet \overline{F_{N-3}} \bullet \overline{F_{N-4}} \bullet F_{N-5} \bullet \overline{F_{N-6}} \bullet \overline{F_{N-7}} \end{matrix}} \quad (18)$$

Above equation shows high logic when the faction part is located in the calibration region that enables the original logarithmic code to be added to $(0.0001)_2$. Otherwise, there is no calibration enabled. The entire calibration block is shown in Fig. 4c with the circuit implementation of (18) and a simplified CLA to achieve the fixed binary calibration that C_{EN} is connected to the F_{N-5} bit. Under this connection, if there is no calibration enabled, the original converted number keeps the same value. Once the calibration is enabled, the original converted number is added with $(0.0001)_2$. Note that the circuit of the case judgement only uses two logic stages no matter how many bits will be converted. For the design of simplified CLA, since F_{N-5} is the last bit in the adder, we can set F_{N-5} as carry-in bit so that the original logarithmic code adds a full zero sequence. Thus, the generate terms in a CLA are all low logic and the propagate terms are equal to the original logarithmic code. We cancel the AND array outputting the generate terms and XOR array outputting propagate terms. We leave only AND array to obtain the carry terms, and XOR array outputs the final results, which are also shown in Fig. 4c. This simplified CLA can reduce two logic stages and the circuit cost due to the proposed calibration.

3.3. Power part

Another important issue is how to drive adiabatic logic using the clocked power. The types of clocked power include sine, trapezoid, and step-by-step waveforms. Sine clocked power can be obtained by oscillators with passive devices to boost oscillation frequency [27]. Trapezoid power can be obtained by the RL circuitry, which brings analogue block to the entire system [28]. Step-by-step power can be obtained by tank capacitors [29]. To save the chip area by removing passive devices, the performance of clocked power built by purely logic gates has been studied [30]. It does reduce the occupied area with low power dissipation but signal integrity degrades when frequency boosts.

In this work, we use the voltage controlled ring oscillator (RO) [31] to build our clock tree. To generate four-phase signals with 90° phase difference, we design four ROs with modified dimensions of transistors since the phase difference is highly related to additive RC delay existed in transistors. We select designing RO with seven stages to balance the oscillation frequency and signal integrity. The entire design of clock part is shown in Fig. 4d. The first stage of this circuit is the control block with diode-connected

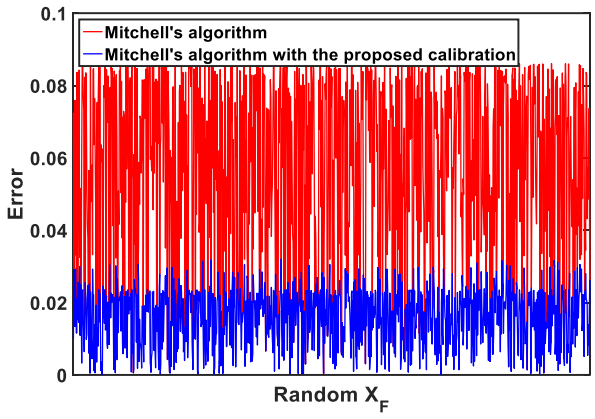


Fig. 5. *Error sweep of the proposed calibration and Mitchell's algorithm*

transistors. The large current going through this control block can boost the oscillation frequency. With the decrease of V_{CTRL}, the oscillation frequency will gradually reduce and the clocked power will finally change to the dc power. The W/L of both pMOS and nMOS transistors with diode connection are set to 800nm/200nm. The W/L of two transistors in the front-end stage of RO is set to 1000nm/200nm, 1600nm/200nm, 2000nm/200nm, and 2400nm/200nm, to build four sine waves with 90° phase difference. For the rest of transistors in this circuit, these are uniformly set to 200nm/200nm.

4. Test results

4.1. Error analysis

The proposed calibration method is developed based on the Mitchell's logarithmic conversion. Fig. 5 shows the error sweep of these two conversions with random fraction. It can be seen that the proposed calibration can obviously supress the conversion error. To evaluate the performance of the proposed calibration in depth, we list a comparison of the proposed one and other reported conversion methods [3, 5-7, 9, 32, 33] in Table 1 in Appendix. The percentage error range can reflect the average error for a given method. The computed percentage error range of our work is 1.55% using Matlab simulation. Thus, using Mitchell's algorithm as the reference, the reduced rate of conversion error in our work is 71.1%, which is larger than reported in [5-7, 9]. On comparison with [33], the reduced rate of conversion error in the proposed work is larger than the case of four fraction regions. To obtain nearly the same error as in [33] for the proposed work, it requires eight fraction regions in [33]. The number of fraction regions is proportional to the circuit complexity and the computation latency. Our calibration method uses three fraction regions to be processed, which is larger than reported in [6, 7, 9] and one case in [32]. However, these three fraction regions are only generated by one case judgement. The calibration block only uses two logic stages for the case judgement as described in the last section. Thus, three fraction regions in this work cannot lower the work speed significantly. Methods in [5, 9, 32] are very time consuming since counter, shifter, or leading-one detector are used, which are not suitable to the fast computation. When the number of fraction regions increases, the computation in

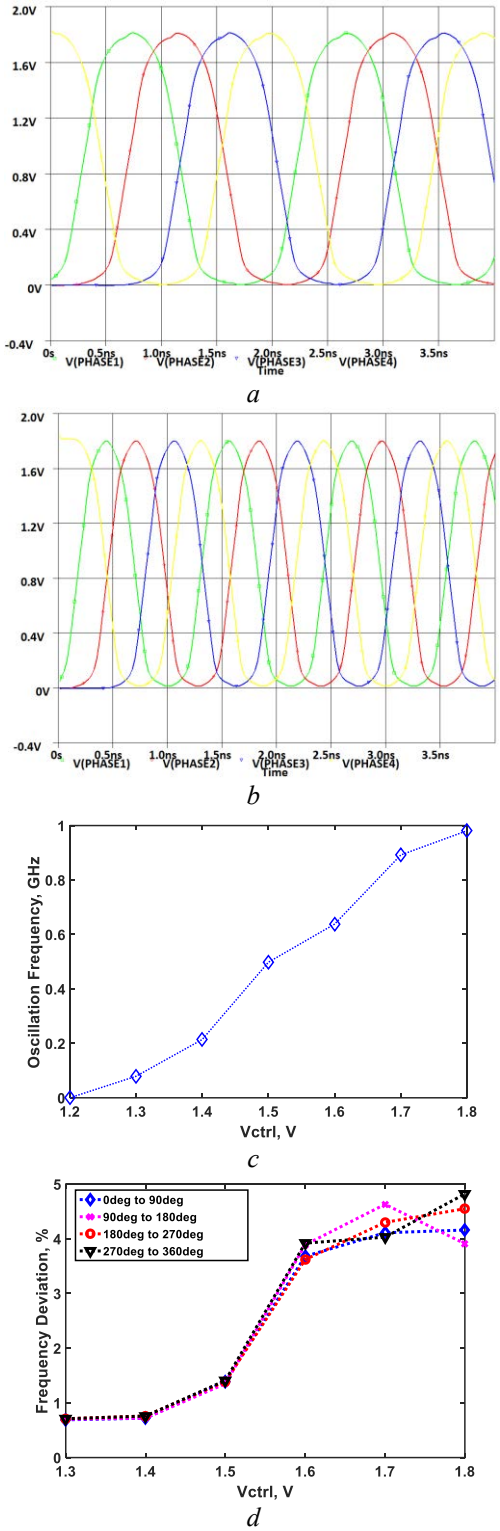


Fig. 6. *Simulations of clock generator*
a Waveform at 498MHz
b Waveform at 0.981GHz
c Oscillation frequency versus control voltage
d Frequency deviation versus control voltage

[32, 33] can reduce further conversion error than in our work. However, both of two works introduce a complex multi-step linear function, which is a penalty on computation speed since each linear function requires multi-bit multiplication and addition to be processed. In [33], it uses LUT to reduce the computation latency of multi-step function at the cost of chip

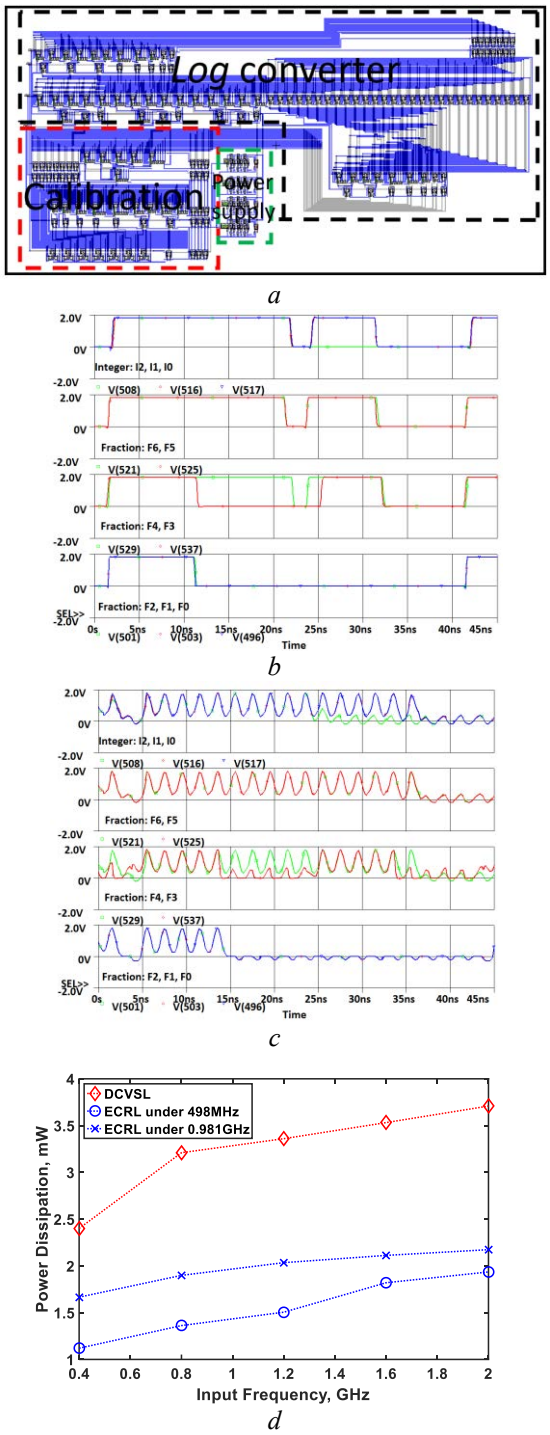


Fig. 7. *Simulations of the entire logarithmic converter*
a Layout view
b DCVSL transient simulation
c ECRL transient simulation
d Comparison of power dissipation under different logic

area. The work in [6, 7] simply uses two fraction regions to speed up the conversion at the cost of conversion accuracy.

4.2. Circuit analysis

We first test the performance of the clock block. Fig. 6a and b show the transient simulation results of 498MHz and 0.981GHz four-phase sine waves under two control voltages, 1.5V and 1.8V, respectively. We can see that the clocked power generated by the proposed circuit is roughly the same as the standard sine wave, which can be used for ECRL work. Fig. 6c shows the oscillation frequency with the variation of V_{CTRL}. We can see the oscillation frequency ranges from 79MHz to 0.981GHz when V_{CTRL} boosts from 1.3V to 1.8V. When the control voltage is lower than 1.2V, the clock block supplies dc power making the following logic block work in high speed DCVSL. Another important issue is the frequency deviation since each two adjacent logic stages must follow 90° phase difference to confirm the correct data transmission in ECRL. Comparing each two ROs outputting two adjacent clocked power, we use 90° phase difference as the standard value. The phase deviation is calculated from the difference between actual phase and 90°, and divided by 90°.

The phase deviation with the variation of oscillation frequency is shown in Fig. 6d. It can be seen that the phase deviation increases with the boost of the oscillation frequency and does not exceed 5% so that the clock part can reliably support the logic part working in ECRL.

Fig. 7a shows the layout of our design using TSMC 180nm CMOS. It includes 1162 transistors with the dimension of 0.063mm^2. Besides some transistors in clock part, the dimensions of the rest of transistors are 200nm/200nm. From the layout, we can see that both calibration block and clock generator do not occupy large chip area. The logarithmic converter is the largest block since we use highly paralleled structure to speed up the conversion instead of counter, leading-one detector and any other blocks, which can process multiple-bit repeatedly using the single circuitry. Fig. 7b and c show the simulation results of the logarithmic codes under DCVSL and ECRL, respectively. The frequency of the clocked power in ECRL transient simulations is 498MHz. The input signal sequence in both DCVSL and ECRL transient simulations are $(11111111)_2$, $(11101000)_2$, $(00001111)_2$, and $(00000000)_2$ at 100MHz. Under these inputs, the outputs are $(111.1111111)_2$, $(111.1110000)_2$, $(011.1111000)_2$, and $(000.0000000)_2$, which are matched to the Mitchell's conversion with the proposed calibration method. The delay in DCVSL is 1.82ns with good signal integrity. In ECRL, the output in the first 5ns is unstable and incorrect since the adiabatic chain supplied by four-phase clocked power outputs the results stage by stage. After 5ns, the output port can correctly transmit the results. Through the sine waves, we can see the distinction of high logic and low logic in ECRL simulations. However, the signal integrity in ECRL is not as good as in DCVSL. This is due to the supplied clocked power, which is not a perfect sine signal as previously shown in Fig. 6a and b. Incomplete charging and discharging are implemented during fast operation of ECRL. In addition, the limited driven-ability of the clock generator in this work degrades the clocked power supply. Fig. 7d shows the simulation results of the power dissipation under both ECRL and DCVSL. We set the frequency of clocked power at 498MHz and 0.981GHz used for our ECRL simulation. The power dissipation of the converter under dc power ranges from 2.4mW to 3.709mW with boost in input frequency. Under clocked power supply, when the frequency of the clocked power is 498MHz, the power dissipation varies from 1.12mW to 1.9352mW. The power dissipation under 0.981GHz clocked power varies from 1.6656mW to 2.1734mW. It can be concluded that ECRL does reduce power dissipation compared to DCVSL. The converter driven

Table 2: Comparison of hardware cost with the prior work and the proposed work

	[5]	[9]	[32]	[6]	[7]	[33]	[12]	This work
8-Bit	200/0/0	72/8/0	202/16/40	46/18/0	10/8/0	64/0/64	0/0/192	120/0/0
16-Bit	400/0/0	144/16/0	404/32/80	16/34/0	20/16/0	128/0/256	0/0/640	343/0/0
32-Bit	800/0/0	288/32/0	808/64/160	32/66/0	40/32/0	256/0/1028	0/0/40k	1198/0/0
64-Bit	1600/0/0	576/64/0	1616/128/320	64/130/0	80/64/0	512/0/4112	N/A	4613/0/0

Table 3: Comparison of maximum path with the prior work and the proposed work

	[5]	[9]	[32]	[6]	[7]	[33]	[13]	This work
8-Bit	30	11	24	16	13	13	N/A	8
16-Bit	54	20	37	24	21	13	N/A	11
32-Bit	102	36	69	40	37	13	640	12
64-Bit	198	68	133	72	69	13	N/A	12

by the low frequency clocked power can save more energy than that driven by the high frequency clocked power.

To evaluate the performance of the proposed work depending on the increased bit length, a comparison is presented in Table 2. It compares our work with the prior work presented in [5-7, 9, 12, 32, 33] from the viewpoint of the number of all bottom cells. Three numbers identified by two slashes, from left to right, represent the number of basic logic cells (all Boolean logic, MUX and transmission gate), unit adders and unit memory cells, respectively. The method to estimate the number of all bottom cells is based on the diagrams of both circuit and system shown in the reported papers. For [32], we used the case of 3-regions to estimate the number of all bottom cells. While in [33], the case of 8-regions is used for the estimation. We observe that when the bit length is 8-bit or 16-bit, the number of all bottom cells in this work is smaller than that in [5, 12, 32, 33]. Over 16-bits, the proposed work and [33] consume much more bottom cells than the other work except [12], in which the conversion is fully implemented by memory array and the usage of memory cells is the largest since all converted results corresponding to the specific binary codes need to be stored. Hardware implementations in [6, 7, 9] is not as sensitive to bit width increasing as in other work. The reason that both [33] and the proposed one consume large number of bottom cells is that the two work use highly paralleled topology instead of the pipelined structure achieved by leading-one detectors or shifters. Thus, to process the input with large bit width with the consideration of the allowable fan-in, in a single stage, the requirement of bottom cells is larger than in the other work using step-by-step computation. Another issue is the conversion latency. A very significant metric to evaluate it is through the maximum path, which requires to determine the number of longest logic stages in entire conversion from the binary code to logarithmic code. Table 3 presents the comparison of maximum path between the proposed work and other prior work [5-7, 9, 13, 32, 33]. It is observed that the proposed one consumes the shortest path to obtain the converted result compared to other designs. The implementations with the computations of pipelined shifting [5, 9, 32] are sensitive to the increase of bit length. The work using PLA [13] is not suitable for the fast conversion. For work in [6, 7], the front-end circuit to achieve multi-step function without LUT contributes the most to the required path. The runner-up in conversion speed is the design in [33], since it also follows the path of highly paralleled topology. The reason that it is slightly slower than the proposed one is that its multi-step function is complex and requires more stages than in the proposed one.

In our design, the latency through the calibration block is not sensitive to the bit width. As mentioned in the last section, the block to implement the case judgement of the calibration block requires only two stages without stage variation as bit width is increasing. Our simplified traditional CLA reduces the number of logic stages required by the calibration. The block outputting fraction number contributes mainly to both required stages and the number of bottom cells with the increase of bit width since one stage has N^2 inputs as shown in Fig. 4b. Other blocks for the enable signals and the integer number do not introduce more stages and hardware cost when bit width increases.

5. Conclusion

In this work, based on Mitchell's algorithm, we propose a novel calibration method to supress the error during logarithmic conversion. The essence of our calibration is to use a fixed code to calibrate a single converted number. At the circuit level, we propose a highly paralleled structure to speed up the logarithmic conversion. The bottom gate is designed by DCVSL to let the system work under both adiabatic logic and fast logic. For the clocked power design, we used four voltage controlled ring oscillators without passive devices to generate four-phase sine signals to drive the converter. This clock block also can be controlled to supply dc power to support traditional DCVSL working. We designed an 8-bit logarithmic converter in layout level to

successfully verify the proposed calibration method. The results prove that when the system work in ECRL, the power dissipation is smaller than that in DCVSL. The performance comparison proves that the conversion latency of the proposed design is not sensitive to the increase in bit width, since computations of shifting and iteration are not introduced into our work.

6. References

[1] Chen, Y. J., Hsu, C. H., Hung C. Y., et al.: 'A 130.3 mW 16-core mobile GPU with power-aware pixel approximation techniques', IEEE Journal of Solid-State Circuits, 2015, 50, (9), pp. 2212-2223.

[2] Chrétien, B., Escande, A. and Kheddar, A.: 'GPU robot motion planning using semi-infinite nonlinear programming', IEEE Transactions on Parallel and Distributed Systems, 2016, 27, (10), pp. 2926-2939.

[3] Mitchell, J. N.: 'Computer multiplication and division using binary logarithms', IRE Transactions on Electronic Computers, 1962, 11, (4), pp. 512-517.

[4] Mahalingam, V. and Nagarajan, R.: 'Improving accuracy in Mitchell's logarithmic multiplication using operand decomposition', IEEE Transactions on Computers, 2006, 55, (12), pp. 1523-1535.

[5] Combet, M., Zonneveld, H. V. and Verbeek, L.: 'Computation of the base two logarithm of binary numbers', IEEE Transactions on Electronic Computers, 1965, (6), PP. 863-867.

[6] Juang, T. B., Chen, S. H. and Cheng, H. J.: 'A lower error and rom-free logarithmic converter for digital signal processing applications', IEEE Transactions on Circuits and Systems II: Express Briefs, 2009, 56, (12), pp. 931-935.

[7] Juang, T. B., Meher, P. K. and Jan, K. S.: 'High-performance logarithmic converters using novel two-region bit-level manipulation schemes', Proceedings of 2011 IEEE International Symposium on VLSI Design, Automation and Test, Hsinchu, Taiwan, April 2011, pp. 1-4.

[8] Kostopoulos, D. K.: 'An algorithm for the computation of binary logarithms', IEEE Transactions on Computers, 1991, 40, (11), pp. 1267-1270.

[9] SanGregory, S. L., Brothers, C., Gallagher, D. et al.: 'A fast, low-power logarithm approximation with CMOS VLSI implementation', Proceedings of 1999 IEEE Midwest Symposium on Circuits and Systems, Las Cruces, USA, August 1999, pp. 388-391.

[10] Huang, S. C., Chen, L. G. and Chen, T. H.: 'The chip design of a 32-b logarithmic number system', Proceedings of 1994 IEEE International Symposium on Circuits and Systems (ISCAS), London, UK, May 1994, pp. 167-170.

[11] Fit-Florea, A., Li, L., Thornton, M. A. and Matula, D. W.: 'A discrete logarithm number system for integer arithmetic modulo 2^{k}: algorithms and lookup structures', IEEE Transactions on Computers, 1994, 141, (5), pp. 281-292.

[12] Wan, Y. and Wey, C. L.: 'Efficient algorithms for binary logarithmic conversion and addition', IEE Proceedings-Computers and Digital Techniques, 1999, 146, (3), pp. 168-172.

[13] Lai, F. S.: 'The architecture and analysis of a hybrid number system processor', Proceedings of 1992 IEEE International Symposium on Circuits and Systems (ISCAS), San Diego, USA, May 1992, pp. 803-806.

[14] Lo, H. Y.: 'Generation of a precise binary logarithm with difference grouping programmable logic array', IEEE transactions on computers, 1985, 100, (8), pp. 681-691.

[15] Abed, K. H. and Siferd, R. E.: 'VLSI implementations of low-power leading-one detector circuits', Proceedings of the IEEE SoutheastCon 2006, Memphis, USA, March 2006, pp. 279-284.

[16] Gok, M., Schulte, M. J. and Arnold, M. G.: 'Integer multipliers with overflow detection', IEEE Transactions on Computers, 2006, 55, (8), pp. 1062-1066.

[17] Abed, K. H. and Siferd, R. E.: 'CMOS VLSI implementation of 16-bit logarithm and anti-logarithm converters', Proceedings of 2000 IEEE Midwest Symposium on Circuits and Systems, Lansing, USA, August 2000, pp. 776-779.

[18] Maksimovic, D., Oklobdzija, V. G., Nikolic, B. and Current, K. W.: 'Clocked CMOS adiabatic logic with integrated single-phase power-clock supply', IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2000, 8, (4), pp. 460-463.

[19] Tenace, V., Calimera, A., Macii, E. and Poncino, M.: 'Quasi-adiabatic logic arrays for silicon and beyond-silicon energy-efficient ICs', IEEE Transactions on Circuits and Systems II: Express Briefs, 2016, 63, (12), pp. 1111-1115.

[20] Kim, C., Yoo, S. M. and Kang, S. M.: 'Low-power adiabatic computing with NMOS energy recovery logic', Electronics Letters, 2000, 36, (16), pp. 1349-1350.

[21] Gutierrez, R. and Valls, J.: 'Low cost hardware implementation of logarithm approximation', IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2011, 19, (12), pp. 2326-2330.

[22] 'Properties of Logarithmic Functions', http://www.montereyinstitute.org/courses/DevelopmentalMath/TEXTGROUP-1-19_RESOURCE/U18_L2_T2_text_final.html, accessed May 2017.

[23] Lee, H. J. and Kim, Y. B.: 'Low power null convention logic circuit design based on DCVSL', Proceedings of 2013 IEEE International Midwest Symposium on Circuits and Systems, Columbus, USA, August 2013, pp. 29-32.

[24] Sathe, V. S., Chueh, J. Y. and Papaefthymiou, M. C.: 'Energy-efficient GHz-class charge-recovery logic', IEEE Journal of Solid-State Circuits, 2007, 42, (1), pp. 38-47.

[25] Rabaey, J., Chandrakasan, A. and Nikolic, B.: 'Digital integrated circuits' (Upper Saddle River, New Jersey, 2003).

[26] Kim, J. M. and Yoo, H. J.: 'Bitwise competition logic for compact digital comparator', Proceedings of 2007 IEEE Asian Solid-State Circuits Conference, Jeju, Korea, November 2007, pp. 59-62.

[27] Teichmann, P.: 'Adiabatic logic: future trend and system level perspective' (Springer Science & Business Media, Berlin, Germany, 2011).

[28] Stoffi, A. B., Amirante, E., Fischer, J., Innaccone, G. and Landsiedel, D.: 'Resonant 90 degree shifter generator for 4-phase trapezoidal adiabatic logic', Advcanced Radio Science, 2003, 1, (9) pp. 243–246.

[29] Nakata, S., Makino, H. and Matsuda, Y.: 'A new stepwise adiabatic charging circuit with a smaller capacitance in a regenerator than a load capacitance', Proceedings of 2014 IEEE International Midwest

Symposium on Circuits and Systems, College Station, USA, August 2014, pp. 439-442.
[30] Zhao, Z., Srivastava, A., Peng, L. and Mohanty, S. P.: 'A low-cost mixed clock generator for high speed adiabatic logic', Proceedings of 2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Pittsburgh, USA, July 2016, pp. 587-590.
[31] Jovanovic, G., Stojcev, M. and Stamenkovic, Z.: 'A CMOS voltage controlled ring oscillator with improved frequency stability', Scientific Publications of the State University of Novi Pazar, Series A: Applied Mathematics, Informatics and mechanics, 2010, 2, (1), pp. 1-9.
[32] Abed, K. H. and Siferd, R. E.: 'CMOS VLSI implementation of a low-power logarithmic converter', IEEE Transactions on Computers, 2003, 52, (11), pp. 1421-1433.
[33] Caro, D. D., Genovese, M., Napoli, E., et al.: 'Accurate fixed-point logarithmic converter', IEEE Transactions on Circuits and Systems II: Express Briefs, 2014, 61, (7), pp. 526-530.

7. Appendix

Table 1 in Appendix section is the comparison between reported conversion methods and this work.

Table 1: Comparison with the reported conversion methods and the proposed calibration

	[3]	[5]	[9]	[32]		[6]	[7]	[33]		Proposed calibration
Number of fraction regions	1	4	2	2	3	2	2	4	11	3
Maximum positive error	0.086	0.0253	0.0292	0.0449	0.0293	0.036	0.0319	0.004	0.004	0.03189
Maximum negative error	0	-0.0062	-0.028	-0.0183	-0.021	-0.009	0	-0.004	-0.004	-0.03197
Error range	0.086	0.0315	0.0572	0.0632	0.0503	0.045	0.0319	0.008	0.008	0.06386
Maximum positive percentage error (%)	5.36	2.293	0.431	0.93	0.431	2.889	2.337	2.93	0.225	1.15
Maximum negative percentage error (%)	0	-0.468	-1.54	-0.554	-0.268	-0.45	0	-1.67	-0.181	-0.4
Percentage error range (%)	5.36	2.761	1.971	1.484	0.699	3.339	2.337	4.6	0.406	1.55
Reduced rate compared to Mitchell's algorithm (%)	0	48.5	63.2	72.3	87	37.7	56.4	14.2	92.4	71.1
Circuit strategy	N/A	Counter; Register.	Shifter Array; Adder.	ROM; Leading-one Detector; Shifter.		Double-Adder.	Logic Array; Adder.	Segment Encoder Array.		Logic Array; Adder.

For all compared metrics in above table, the computation methods are listed as follows:

1) Maximum positive and negative errors are two extreme values and can be found through the error curve directly.

2) The error range can be obtained by the difference between the maximum positive error and maximum negative error.

3) The percentage error range is obtained through $\varepsilon_{Mitchell}$-ε_{NEW}, where $\varepsilon_{Mitchell}$ is the mean value of the error using Mitchell's conversion, ε_{NEW} is the mean value of the error using the novel conversion method. Specifically, for the proposed work, ε_{NEW} can be calculated from (8) in Section 2. Using the same equation, if we only use positive part in the error curve for the integral, the maximum positive percent error can be obtained. Therefore, if the integral is applied to the negative part in the error curve, the maximum negative percentage error can be obtained.

4) The reduced rate compared to Mitchell's algorithm is calculated through $(PER_{Mitchell}\text{-}PER_{NEW})/PER_{Mitchell}$, where PER is the percentage error range.