Automatic Synthesis of Comparator Circuit Using Genetic Algorithm

Takayuki NEGISHI†, Naoki ARAI, Nobukazu TAKAI,

Masato KATO, Hiroaki SEKI, Sumit Kumar BISWAS and Haruo KOBAYASHI

Gunma University, 376-8515 Japan

E-mail: †t13801472@gunma-u.ac.jp ; takai@gunma-u.ac.jp

Abstract This paper describes our challenge of automatic analog circuit design that the genetic algorithm chooses the optimal circuit topology and HSPICE optimizing function obtains their optimal parameter values automatically, by focusing on a comparator circuit which is one of the important analog building blocks. Automatic design for analog circuit has not been realized yet, even though automatic design is being used in digital circuit design; the reason behind this is that the number of parameters to be considered in an analog circuit design is much larger than digital circuit design. However nowadays it is extremely difficult to design ICs manually due to their large scale integration and hence their automatic design is demanded. We present our automatic circuit design flowchart programmed with Java language which realizes 1-click automatic synthesis of the comparator, and shows that our method can obtain a better performance comparator compared to an initially-set comparator.

Keywords Automatic Synthesis, Analog Circuit, Comparator, Genetic Algorithm, Optimization

1. Introduction

Scaling of integrated circuits is advancing day by day, and it is becoming difficult to design them manually. In recent years, automatic design is being used in digital circuit design. However, automatic design for analog circuit has not been realized yet. The reason behind this is that the number of parameters to be considered in the analog circuit design is much larger than that of digital circuit. Consequently, analog circuit design requires skilled designers and relatively long design time.

Various methods for automatic analog circuit design have already been proposed in recent years [1][2][3][4][5]. However most of them set one circuit topology *a priori* and choose their optimal parameter values with the genetic algorithm [6]. Our study here is to develop an algorithm to choose the optimal circuit topology automatically using the genetic algorithm and optimize their parameter values using HSPICE optimizing function. This algorithm is challenging so that we focus on a simple-yet-important analog circuit block, a comparator [7], as our automatic design circuit example.

We show our circuit design flowchart which we have implemented with Java-language programming; we have realized its 1-click automatic synthesis. Our automatic synthesis results show that our algorithm can obtain a better performance comparator circuit compared to an initially-set comparator circuit.

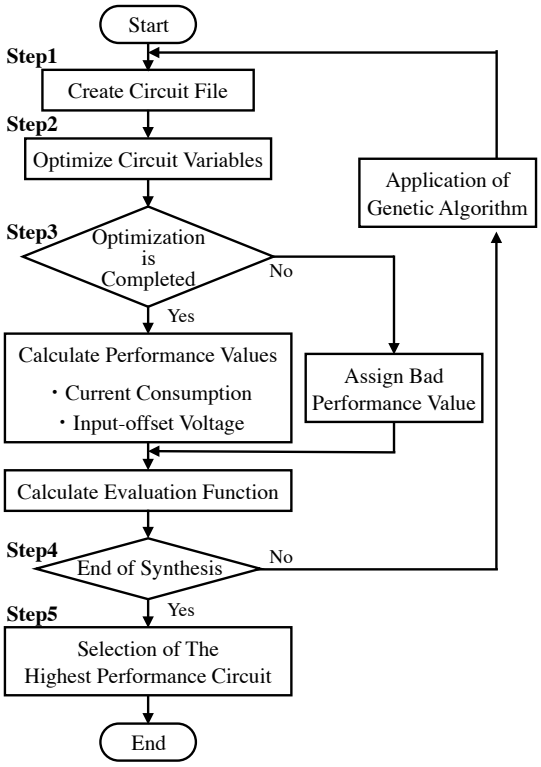

Figure 1: Our automatic comparator circuit synthesis flowchart.

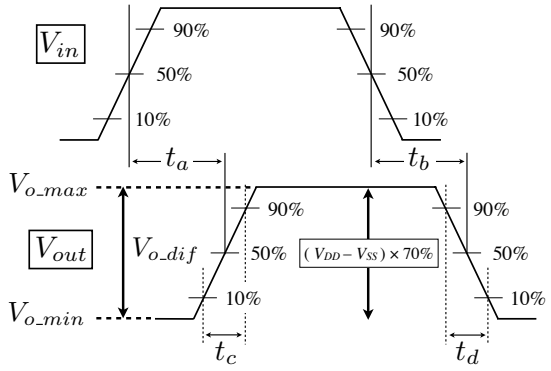

Figure 2: Comparator response time parameters.

Table 1: Comparator performance evaluation items and target values.

Evaluation Item	Target Value
Output voltage difference	$\geq$ 2.0V
Propagation delay time	$\leq$ 168ns
Output voltage transition time	$\leq$ 144ns
Current consumption	$\leq$ 5.65mA
Input-offset voltage	$\leq$ 59.6mV

2. Method of Automatic Synthesis

2.1. Flow of Automatic Synthesis

Fig. 1 shows the flow of our automatic synthesis, and we have implemented its processing chain with Java-language programming (approximately 2,500 lines). The explanation of the flow is as follows:

1. First, our program creates one file that describes the circuit information. The circuit file uses HSPICE input file format and prepares for variable parameters to which HSPICE optimizing function is applied.

2. Next, our program determines proper values of parameter variables based on specification using the HSPICE optimizing function.

3. - If the performance optimization can reach the demanded design specification (judged from the comparator output voltage difference that is between V_{o_max} and V_{o_min} in Fig. 2) , our program determines performance values by calculating such as current consumption and input-offset voltage.
   - Else if the performance optimization cannot reach the demanded design specification with a certain circuit topology, the circuit topology is considered as "bad" and its evaluation value is rated to a very low figure. Then we skip the analysis part in the flowchart to save computation time.

4. Increment loop number by 1.
   - If the loop number is less than the specified value, then choose another comparator topology using the genetic algorithm and go to Step 1.
   - Else if the loop number is equal to the specified value, go to step 5.

5. Select the circuit with the maximum evaluation value as the highest performance circuit, among all of the evaluated circuits above.

2.2. Result Evaluation

We consider here to evaluate one circuit performance (labeled as "k" with one figure, evaluation value E_k). Overall evaluation of the circuit is represented by an evaluation function (or its calculation result as a fitness value F in eq. (1)) which is product of all evaluation values E_k's for each circuit performance.

$$F = \sum_{k=1}^{N+M} E_k \tag{1}$$

We set the evaluation value calculation expressions to eq. (2) and eq. (3), where s_k represents k-th performance value (simulation result) while t_k represents its target value $(k = 1, 2, .., N + M)$.

For the items (such as output voltage difference) aiming *higher* than the target value, k-th evaluation value is calculated using eq. (2) by substituting k-th performance value s_k and target value t_k

$$E_k = \begin{cases} \frac{s_k}{t_k} & (\text{in case } s_k \leq t_k) \\ 1 + \log\left(\frac{s_k}{t_k}\right) & (\text{in case } s_k > t_k). \end{cases} \tag{2}$$

We consider that when s_k becomes *greater* than t_k, the corresponding performance labeled as k reaches to a satisfactory level and E_k uses the second expression in eq. (2) to suppress the increment of the evaluation value E_k for s_k. This can solve the trade-off problem of having only one optimized 'good' value, while the others remain 'bad'; design trade-off exists in the circuit performance, and therefore there is every bit of chance of deteriorating one item while improving the other one but the above method can solve this problem.

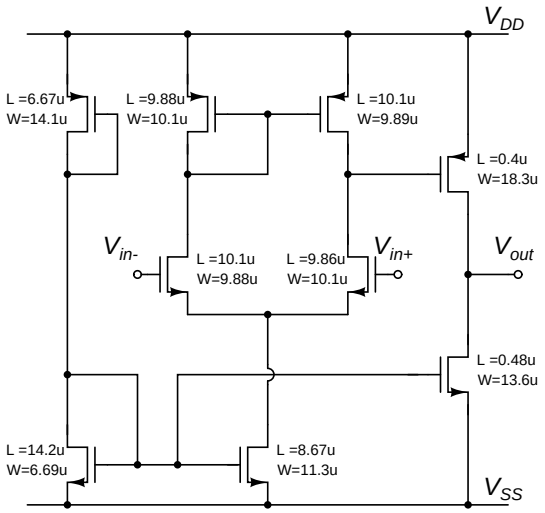

Figure 3: Initial comparator circuit.

For the items (such as propagation delay time, output voltage transition time, current consumption, input-offset voltage) aiming $lower$ than the target values, k-th evaluation value is calculated using eq. (3) by substituting k-th performance value s_k and target value t_k $(k = N+1, N+2, .., N+M)$:

$$E_k = \begin{cases} \frac{t_k}{s_k} & (\text{in case } s_k \geq t_k) \\ 1 + \log\left(\frac{t_k}{s_k}\right) & (\text{in case } s_k < t_k). \end{cases} \quad (3)$$

Similarly we consider that when s_k becomes $less$ than t_k, the corresponding performance labeled as k reaches to a satisfactory level and E_k uses the second expression in eq. (3).

Table 1 shows the evaluation items and target values as well as the fitness value F. The calculation test bench of each item is referred to the comparator data sheet in industry [8]. Target values use performance values of the initial comparator circuit in Fig. 3.

2.3. Application of Genetic Algorithm

Our program searches for comparator circuit topology using genetic algorithm and with the initial circuit in Fig. 3. It decides the connections among MOSFETs terminals and changes a $gene$ (or comparator circuit topology) using 'Selection' and 'Mutation' operations in the genetic algorithm, There are some operation methods of 'Selection' in the genetic algorithm, and here we have used 'Fitness proportion selection'. We do not use 'Crossover' (Crossover probability is 0%) in the genetic algorithm this time, because it may create floating nodes inside the circuit. Next we will explain operations of 'Selection' (Fitness proportion selection) and 'Mutation'.

2.3.1. Fitness Proportion Selection

This section explains 'fitness proportion selection'. Let the fitness value of circuit topology k $(k = 1, .., N_p)$ in certain generation be F_k, and then eq. (4) shows the expectation that the individual k is chosen.

$$\frac{F_k}{\Sigma_k F_k / N_p} = \frac{F_k}{\bar{F}}. \quad (4)$$

Here N_p represents the population and $\bar{F}$ represents the mean of the fitness value. Our program generates next-generation population (circuit topologies) based on eq. (4); for example, the circuit topology k is selected as the next-generation population with the probability proportional to $F_k/\bar{F}$; this is called as 'fitness proportion selection' in the genetic algorithm theory [6].

2.3.2. Mutation

Our program operates 'Mutation' to make a different circuit topology, where we take care of not producing floating nodes inside the circuit. Purpose of mutation process is explained below:

Let us consider the circuits in Fig. 4. There is node-A (in the upper circuit) where two elements from the high potential side and also two elements from the low potential side are connected, which is defined as an H-type node. Our program may mutate the H-type node to split into two nodes as shown in the lower circuits in Fig. 4.

Next consider the circuits in Fig. 5, where the upper circuit has two H-type nodes; node-B and node-C. Our program may mutate to create new nodes (node-D, node-E) and reduce the number of vertically connected MOSFETs.

Then look at the circuits in Fig. 6, where in the upper circuit, node-F splits a current path into two paths, and node-G gathers two current paths as one path. Our program may mutate to connect node-F to another node as shown in the lower circuits.

In addition, our program may mutate to change not only the node but also the type of MOSFET; our program may change PMOS to NMOS and vice versa; probabilities of both are 50% and 50%.

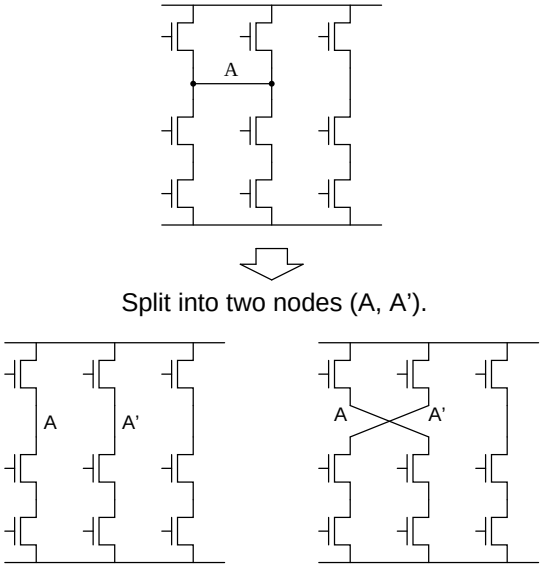

Figure 4: Example of mutation (case 1). The above circuit stays as itself or it may become the lower left or right circuit in the next generation.

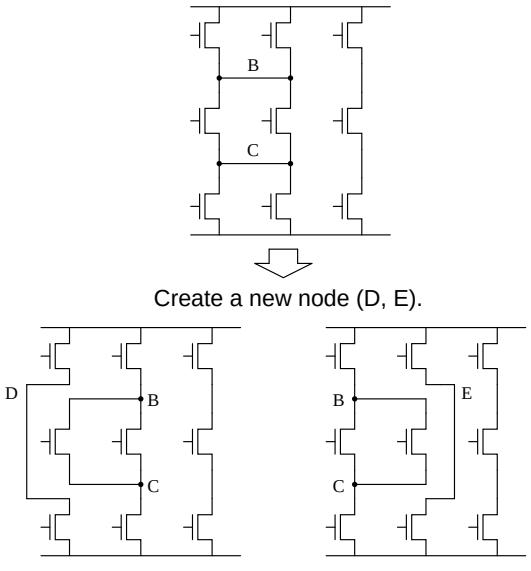

Figure 5: Example of mutation (case 2).

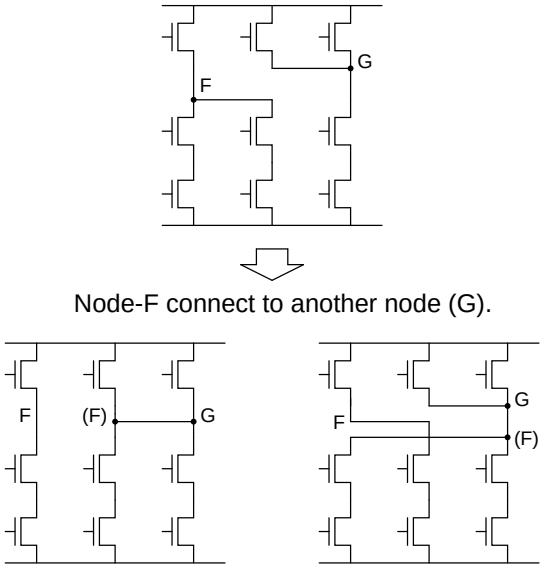

Figure 6: Example of mutation (case 3).

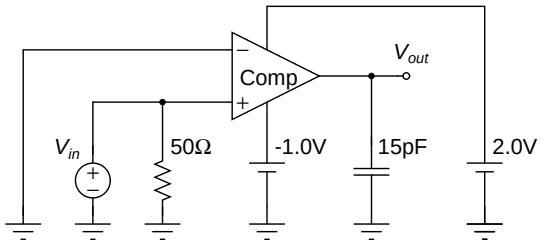

Figure 7: Test bench for propagation delay time, output voltage transition time and current consumption calculations.

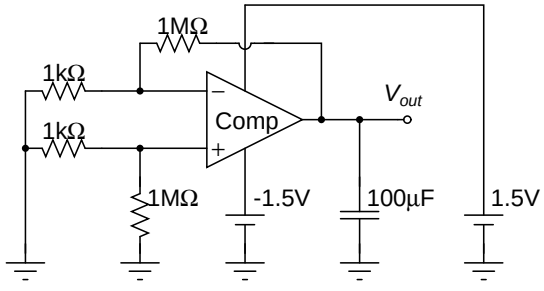

Figure 8: Test bench for input-offset voltage calculation.

2.4. Initial Circuit to Apply Genetic Algorithm

Genetic algorithm result depends largely on the initial value; here the initial circuit is a basic comparator circuit in Fig. 3. In our all simulations, we use 0.18 μm CMOS process parameters.

2.5. Performance Value Calculation Circuit

This section shows test bench of evaluation items. Test bench of Fig. 7 calculates output voltage difference, propagation delay time, output voltage transition time and current consumption (during these testings, V_{in-} is connected to GND (Note that V_{DD} is plus supply and V_{SS} is minus supply.)), whereas test bench of Fig. 8 calculates input-offset voltage. Furthermore, 4 different response time parameters are defined as shown in Fig. 2; propagation delay time is defined as $(t_a + t_b)/2$ and output voltage transition time is defined as $(t_c + t_d)/2$. Current consumption defines effective current value in one cycle of input pulse and Input-offset voltage is defined by output terminal voltage.

3. Result of Automatic Synthesis

Our program executes the automatic synthesis under the conditions in Table 2. It generates thirty circuits in one generation. All the circuits of the first generation use the initial circuit topology. In the second and later generations, there can be many circuit topologies in one generation, and some of them are succeeded in the next generation using 'Selection' algorithm and some circuit topologies may change using 'Mutation'; it continues up to 200 generations.

Fig. 9 shows the transition of the maximum fitness of each generation; fitness upgrades until about 25 generations, but there are no significant changes later on. Fig. 10 shows best performance circuit among all automatically synthesized circuits. Fig. 11 shows the input and output waveform of the best performance circuit. Fig. 12 and Fig. 13 shows the enlarged views of the input and output waveforms. These figures show that the synthesized comparators work. Table 3 shows performance values of the circuit in Fig. 10, and we see that performance of the synthesized circuit is better than that of the initial one.

Table 2: Genetic algorithm conditions for automatic synthesis.

Item	Value
Population (N_p)	30
Generation	200
Crossover rate	0%
Mutation rate	30%

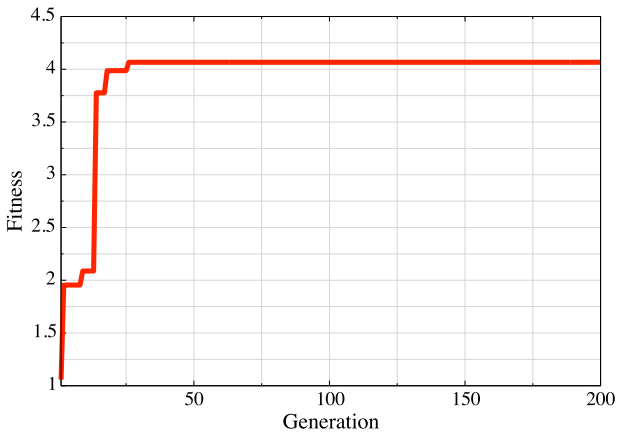

Figure 9: Transition of the maximum fitness value F (which is product of all the evaluation values E's).

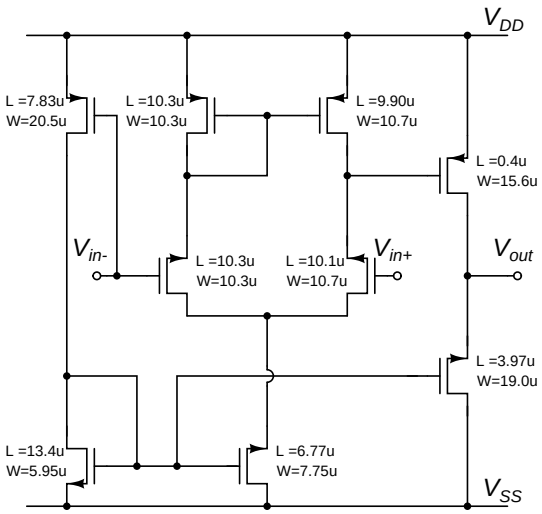

Figure 10: The best performance comparator circuit among all automatically synthesized circuits.

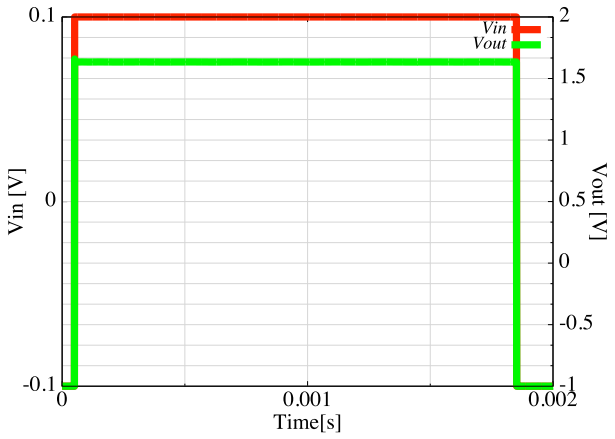

Figure 11: Input and output waveforms of the best performance comparator circuit in Fig. 10.

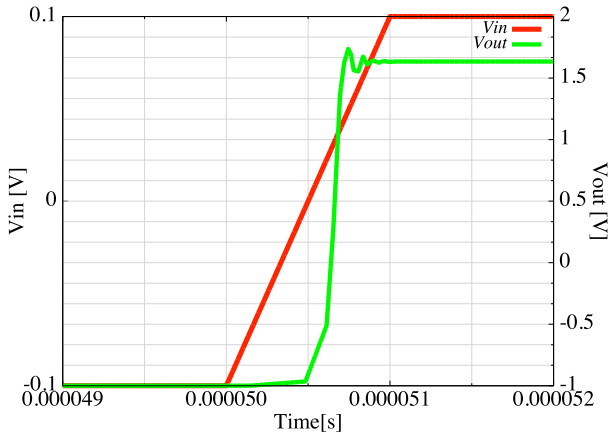

Figure 12: Input and output waveforms (enlarged view of their rising parts in Fig. 11) .

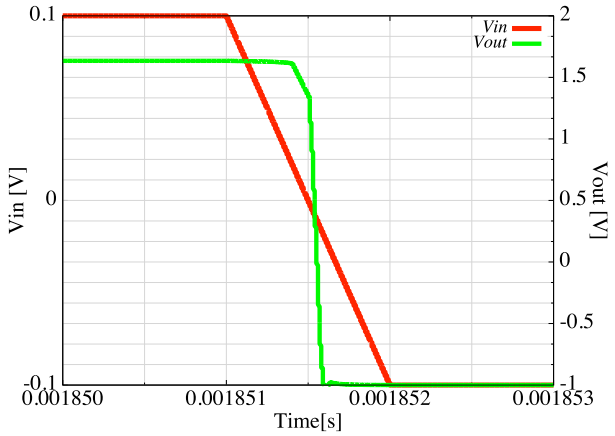

Figure 13: Input and output waveforms (enlarged view of their falling parts in Fig. 11).

Table 3: Target values and simulation results.

Evaluation Item	Value Type	Initial Circuit	Final Circuit
Output voltage difference	Performance	2.29V	2.72V
	Evaluation	1.06	1.13
Propagation delay time	Performance	168ns	73.7ns
	Evaluation	1.0	1.36
Output voltage transition time	Performance	144ns	272ns
	Evaluation	1.0	0.72
Consumption current	Performance	5.65mA	1.48mA
	Evaluation	1.0	1.58
Input-offset voltage	Performance	59.6mV	3.0mV
	Evaluation	1.0	2.3
Fitness value (product of all evaluation values)		1.06	4.06

4. Conclusion

In this study, we have developed automatic synthesis program to design a comparator circuit using genetic algorithm. Our automatic synthesis shows that the performance of the synthesized circuit is better than that of the initial circuit and its program runtime for automatic synthesis was about 20 hours on a standard PC. We have confirmed that the evaluation value increases as the generation repeats compared to the initial circuit. We have used the HSPICE optimizing function to determine circuit variable parameters as well as genetic algorithm to determine the circuit topology.

Future study challenge is to realize successful automatic comparator circuit synthesis from its *faulty* initial circuit, as well as its silicon proof.

References

[1] K. Jin'no. An Automated Circuit Design Procedure by Means of Genetic Programming. *International Symposium on Nonlinear Theory and its Applications*, pages 194–197, Oct 2005.

[2] J. Yu and Z. Mao. A Design Method in CMOS Operational Amplifier Optimization Based on Adaptive Genetic Algorithm. *WSEAS Transactions on Circuits and Systems*, vol.8:548–558, July 2009.

[3] N. Unno, S. Takagi, and N. Fujii. Design Automation of Analog Circuits by Combination of Circuit Blocks: Synthesis of OpAmp. *The Institute of Electrical Engineers of Japan*, ECT-04-18:35–40, Jan 2004.

[4] N. Kitamura and N. Takai. Optimizing method for analog circuit design using immune algorithm. *The Institute of Electrical Engineers of Japan*, ECT-05-45:37–42, June 2005.

[5] N. Arai, N. Takai, B. S. Kumar, and H. Kobayashi. Automatic Design of Analog Filter Using Genetic Algorithm. *4th International Conference on Advanced Micro-Device Engineering,* Kiryu, Japan, Dec 2012.

[6] D. E. Goldberg and K. Sastry. *Genetic Algorithms: The Design of Innovation, 2nd edition*. Springer, 2010.

[7] R. J. Baker. *Mixed-Signal Circuit Design, Second-Edition*. Wiley, 2002.

[8] Renesas Electronics. *HA1631S01/02/03/04*. http://documentation.renesas.com/doc/products/linear/rjj03d0044_ha1631s01.pdf, 2006.