Clock-Deskew Buffer Using a SAR-Controlled Delay-Locked Loop

Guang-Kaai Dehng, *Student Member, IEEE*, June-Ming Hsu, Ching-Yuan Yang, *Student Member, IEEE*, and Shen-Iuan Liu, *Member, IEEE*

***Abstract*—A successive approximation register-controlled delay-locked loop (SARDLL) has been fabricated in a 0.25-μm standard n-well DPTM CMOS process to realize a fast-lock clock-deskew buffer for long distance clock distribution. This DLL adopts a binary search method to shorten lock time while maintaining tight synchronization between input and output clocks. The measured lock time of the proposed SARDLL is within 30 clock cycles at 100-MHz clock input. The power dissipation is 3.3 mW (not including off-chip driver's) at a 1.1-V supply voltage while the measured rms and peak-to-peak jitter are 11.3 ps and 95 ps, respectively.**

***Index Terms*—Clock skew, lock time, PVTL, SARDLL, static phase error.**

I. Introduction

With the rapid advances in semiconductor technologies, modern digital systems operated at several hundred megahertz have been successfully developed for many years. Since there are more and more IC modules integrated on the same printed-circuit board (PCB), the clock-skew problem will undoubtedly be significant and becomes one of the bottlenecks for high-performance systems. The clock-skew problem exists in several different situations. For example, the input clock driver in any chip will somewhat introduce uncertain time delay between the internal clock and the external clock. Thus, the internal clocks in a multi-chip system become asynchronous and problems will occur when data transfer between chips is needed. This phenomenon will also become more serious for chips operated at gigahertz in the future. In addition, on a large PCB, the length of traces between different chips and the clock generator may differ from each other. Hence, the clock-skew problem will inevitably exist. Similar problems also happen in a multi-board system in which different boards are connected through cables. Briefly speaking, the clock-skew problem comes from different propagation delays of system clocks on the board or in a chip, and is usually dependent on process, voltage, temperature, and loading (PVTL), which make it a complicated issue.

Manuscript received October 26, 1999; revised February 1, 2000. This work was supported by the National Science Council under Grant NSC89-2215-E-002-024.

G.-K. Dehng, C.-Y. Yang, and S.-I. Liu are with the Department of Electrical Engineering, National Taiwan University, Taipei 106, Taiwan, R.O.C. (e-mail: lsi@cc.ee.ntu.edu.tw).

J.-M. Hsu was with the Department of Electrical Engineering, National Taiwan University, Taipei 106, Taiwan, R.O.C. He is now with the Electronics Research and Service Organization, Industrial Technology Research Institute, Hsinchu 310, Taiwan, R.O.C. (e-mail: jmhsu@erso.itri.org.tw).

Publisher Item Identifier S 0018-9200(00)06432-5.

Reducing the clock skew can not only further increase system clock frequency but also avoid system malfunction. Phase-locked loops (PLL's) and delay-locked loops (DLL's) have been widely adopted to solve the clock-skew problem. Such kinds of circuits are called clock-deskew buffers. A DLL consists of a phase detector (PD) or a phase comparator (PC), a variable delay line, and a controller to convert the PD's output signal to a control signal for the delay line. It detects the phase error between the input clock and its output clock and automatically tunes the delay line to insert an optimal delay time between them for clock synchronization.

The design of DLL's can be classified into two types: analog and digital. The delay line in an analog DLL is controlled through a loop filter, as shown in Fig. 1 [1]. Two off-chip transmission lines are involved in the clock-deskew system. One connects the clock-deskew buffer's output node with one of the PC's input nodes. The other connects the clock-deskew buffer's output node with the remote chip. Assume that the two transmission lines, as well as the clock input buffers in front of the PC, are identical. A parameter called *loop delay* [2] can be defined as the time delay for the input clock signal to propagate through the input clock buffer, the variable delay line, the output clock buffer, and the feedback transmission line. It can be given as

$$T_{\text{loop}} = T_{\text{input}} + T_{\text{delay}} + T_{\text{output}} + T_{\text{fb}} \tag{1}$$

where T_{input}, T_{delay}, T_{output}, and T_{fb} denote the delay time of the input clock buffer, the variable delay line, the output clock buffer, and the feedback transmission line, respectively. Another propagation delay, T_{forward}, is defined as the time delay for the input clock signal to propagate through the input clock buffer, the variable delay line, the output clock buffer, and the forward transmission line. It is given as

$$T_{\text{forward}} = T_{\text{input}} + T_{\text{delay}} + T_{\text{output}} + T_{\text{fw}} \tag{2}$$

where T_{fw} is the clock delay time of the forward transmission line. As long as the input capacitance of the clock-deskew buffer and the remote chip are close to each other, T_{fw} is close to T_{fb}. In other words, T_{forward} is close to T_{loop}. When the DLL is locked, T_{loop}, as well as T_{forward}, becomes an integer multiple of the clock period and the two input signals of the PC are synchronous. At the same time, the remote clock will also coincide with the input clock.

Since the delay line in an analog DLL is adjusted in a continuous manner, the analog approach can result in smaller static

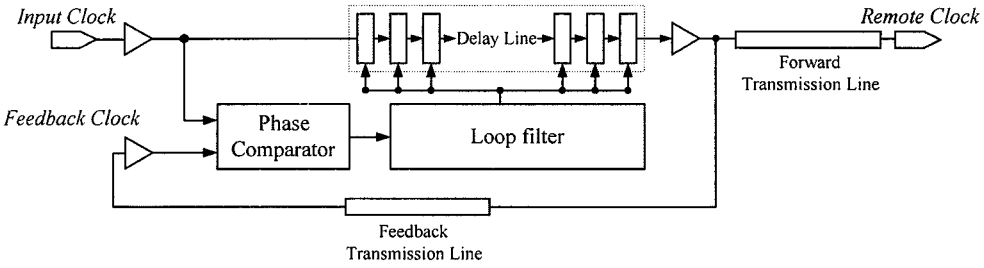


Fig. 1. Analog DLL.

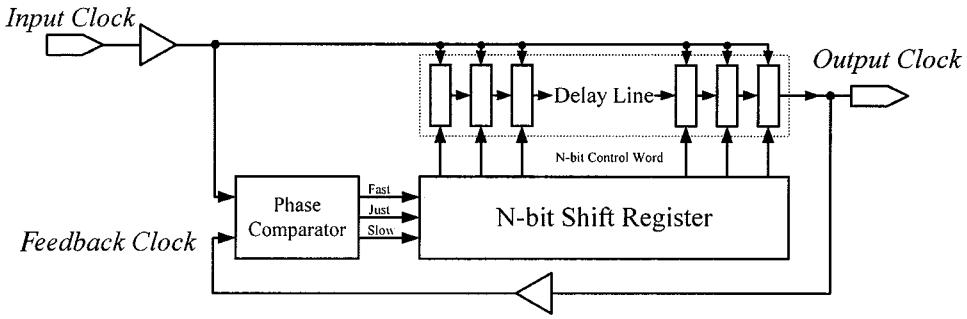


Fig. 2. Register-controlled DLL.

phase error than the digital approach. Besides, low-power operations and small chip area can be achieved in an analog DLL [3]. However, it is not suitable for future low-voltage applications because it cannot provide enough delay range under low supply voltages. Moreover, it is more susceptible to process variations and less immune to power-supply noise. Contrarily, the digital approach [1], [2], [4]–[6] can provide more robust operations over power-supply noise and PVTL effects. Besides, it has low standby current consumption and can exhibit shorter lock time than the analog one at the expense of inherent quantization phase error.

In this paper, a low-voltage all-digital DLL-based clock-deskew buffer is designed in a 0.25-μm standard n-well double-poly double-metal (DPTM) CMOS process for long distance clock distribution. Two conventional architectures of digital DLL's will be reviewed and a successive approximation register-controlled DLL (SARDLL) will first be discussed in Section II. Section III describes the building blocks of the proposed SARDLL. Experimental results are presented in Section IV and Section V gives the conclusions.

II. Conventional Digital DLL's and SARDLL

Fig. 2 shows the block diagram of the register-controlled DLL (RDLL) [1], [4], [5]. The feedback clock signal is the delayed version of the input clock signal, and the shift register controls the amount of the delay time. The PC compares the phases of the input clock signal and the output clock signal. The outputs of the PC, *Fast*, *Just*, and *Slow*, are used to control the shift register. The input clock signal is a common input for every delay stage. At any time, only one bit of the shift register is active to select a point of entry of the delay line for the input clock signal. The number of the delay stages which the input clock signal goes through determines total amount of delay. When *Fast* is active, the feedback clock leads the input clock, and the high bit in the shift register will be shifted left to increase the delay time. When *Slow* is active, the situation is opposite and the high bit in the shift register will be shifted right to decrease the delay time. When *Just* is active, the phase error between the input clock and the output clock is within one unit delay, and the data in the shift register will be held. Under this circumstance, the loop is locked and will not alter until the phase error exceeds the unit delay again. The resolution of the RDLL is determined by the unit delay of the delay line and the total delay time of the delay line determines the DLL's deskew range and the lowest operation frequency. Wider deskew range or lower operation frequency can be achieved by adding more delay stages in the delay line. However, larger chip area would be the penalty.

In order to solve the problem stated above, a counter-controlled DLL (CDLL) has been proposed [2], [6]. Fig. 3 shows the block diagram of the CDLL. It is similar to the RDLL, except that an up/down counter substitutes for the shift register to control the delay line. In addition, the delay line is designed in a binary-weighted manner and no longer consists of delay stages with equal delay time. The n-bit control word from the up/down counter in Fig. 3 determines whether the input clock goes through the delay stage or just passes it. The principle of operation is similar to that case in a RDLL except that the n-bit up/down counter counts up or down to control the delay line. Compared with the RDLL, if 64 delay stages are required in a RDLL, only 6 delay stages are required in a CDLL. Besides, the 64-bit shift register in a RDLL can be replaced by a 6-bit up/down counter. Assume that the 64-bit shift register in a RDLL and the 6-bit up/down counter in a CDLL occupy equal chip area. If both of RDLL and CDLL achieve fine resolution through interpolation, then the delay line of RDLL will have larger offset delay time than the CDLL ($64 > 6$) and occupy larger chip area while the tuning ranges are the same. Using a CDLL, the chip area of the DLL could possibly be reduced while maintaining the same deskew range and having the same limitation for low-frequency operation as in a RDLL.

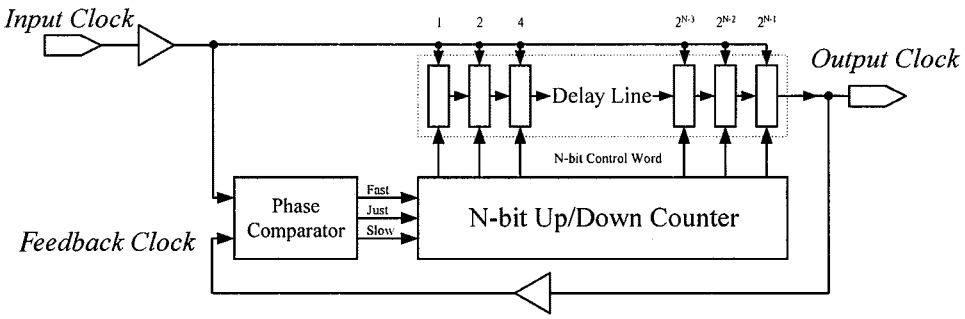


Fig. 3. Counter-controlled DLL.

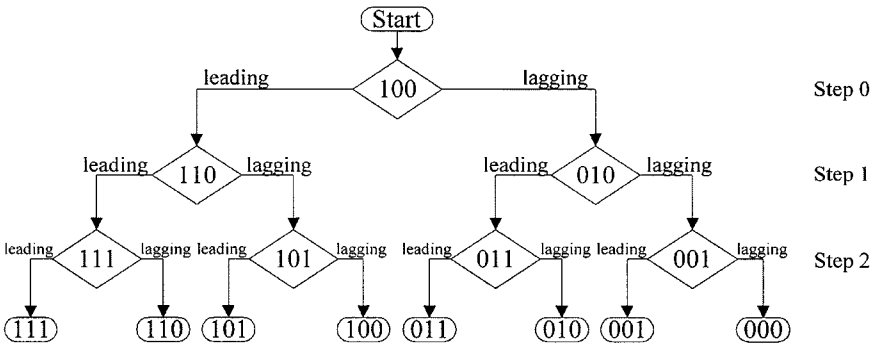


Fig. 4. Flowchart for weighing sequence.

Besides the cost of chip area, another important parameter to evaluate the performance of DLL's is the lock time. In the case of the RDLL, if the point of entry of the delay line is chosen in the middle at first, the longest lock time will be 32 input clock periods for the 64-stage delay line. In the case of the CDLL, if the initial value of the counter is set as 32, the longest lock time will also be 32 input clock periods when a 6-bit up/down counter is used. Both of the digital DLL's mentioned above exhibit the same lock time. Shorter lock time is preferable, especially in the design of high-speed memory applications. A principle called synchronous mirror delay (SMD) has been proposed [7] to achieve fast-lock operation within two input clock periods. More or less, it behaves like a measure-controlled DLL [1]. However, tight synchronization between input clock and output clock is not provided in this case since SMD is an open-loop system, not a closed-loop system.

The operation mechanism of the general DLL's can be treated as a searching sequence. The closed loop tends to search for an optimal delay and tries to insert it between the input clock and output clock. From this point of view, the binary search algorithm can be applied to reduce the searching time. The flowchart in Fig. 4 illustrates the algorithm when a 3-bit binary-weighted delay line is used [8]. In the beginning, the most significant bit (MSB) of the control word is set to 1, and all the other bits are set to 0. The PC examines whether the output clock leads the input clock or not. If so, the MSB remains high. If not, it is set to be low and held constant. In this way, the MSB is determined. The process is then repeated for each following bit until the least significant bit (LSB) is determined. The binary search algorithm is the main operation principle of the proposed SARDLL. Theoretically, when a 6-bit binary-weighted delay line is used, a SARDLL can achieve lock time of six clock periods, which is faster than a RDLL or a CDLL. In this paper, the SARDLL is developed to shorten the lock time while maintaining tight synchronization for the purpose of long distance clock distribution.

III. Circuit Description

The block diagram of the SARDLL is shown in Fig. 5. It consists of a PC, a digital-controlled delay line as in the conventional digital DLL's, and an additional frequency divider, an initial circuit (INCKT), and a 6-bit SAR [8], [9] to provide a control word for the delay line. Again, two off-chip transmission lines are involved in the clock-deskew system. Each block in the SARDLL is described as follows.

A. Digital-Controlled Delay Line

Fig. 6(a) shows the entire binary-weighted digital-controlled delay line. The number on the top of each stage denotes the number of unit delay it provides. The delay line is divided into two sections, *coarse* section and *fine* section. The coarse section consists of 7 delay stages, and each stage provides 8 unit delays. Instead of combining the stages controlled by b_4 or b_5 into two stages with longer delay time, the 16-unit delay stage (controlled by b_4) is composed of two 8-unit delay stages and the 32 delay stage (controlled by b_5) is composed of four 8-unit delay stages. This is due to fact that the longer the delay time per stage is, the more difficult it is for the rise time and fall time of the delay stage to match with each other. Moreover, delay stages with longer delay time will suffer from larger variations in the clock duty cycle, especially in the case of low supply voltages. To avoid such potential variations, more stages with equal delay time are cascaded in the design at the cost of the slightly increased chip area.

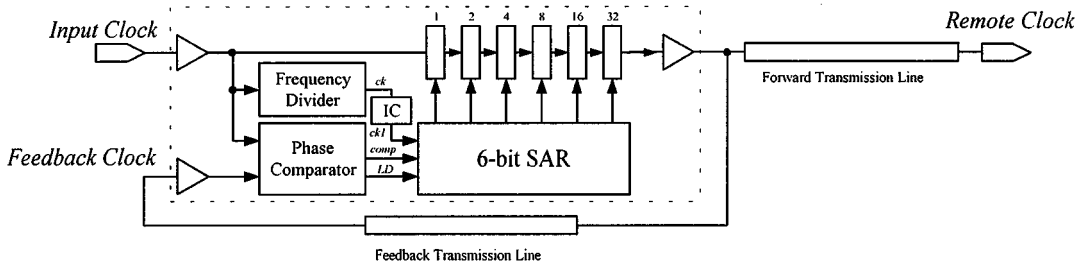


Fig. 5. SAR-controlled DLL clock-deskew buffer.

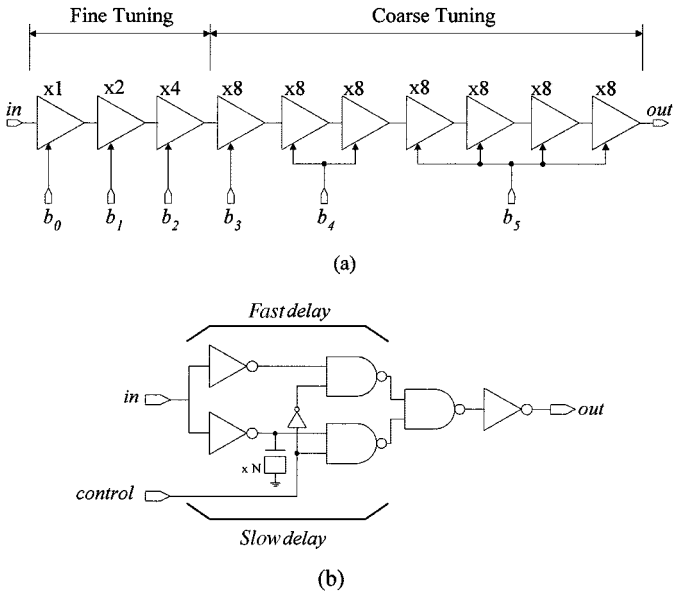


Fig. 6. (a) Binary-weighted digital-controlled delay line. (b) Schematic of each delay stage employing interpolation.

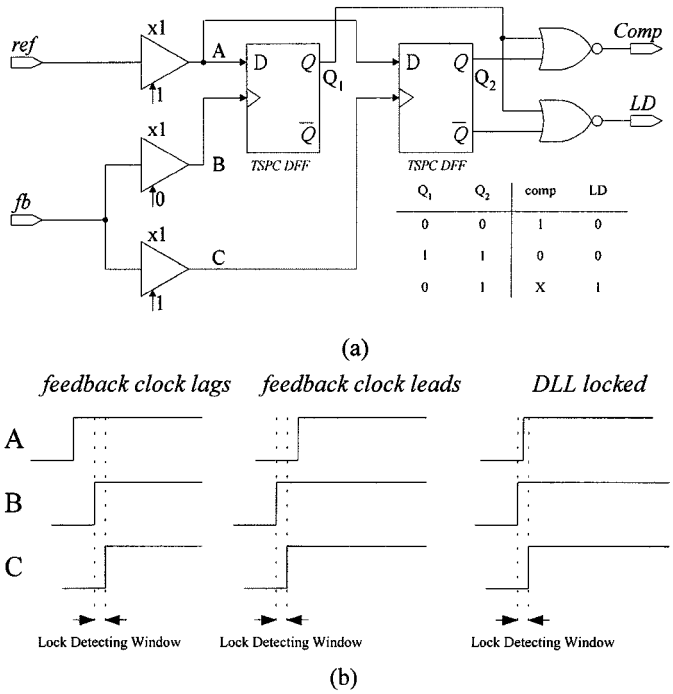


Fig. 7. (a) Schematic of the PC. (b) Operation principle.

The schematic of each delay stage is shown in Fig. 6(b). In the design of delay stages, one of the important things is to preserve 50% clock duty cycle. Usually, the propagation delay through a delay stage resulting from a HIGH-to-LOW transition is not equal to that of a LOW-to-HIGH transition. This nuisance inevitably will result in the undesired pulse shrinkage and duty cycle distortion of the output clock. Here, the concept of symmetrical delay stages is very similar to that in [5]. In order to achieve high resolution in a low-voltage digital DLL, the unit delay is obtained from the difference of the delay time of the two different signal paths in Fig. 6(b). When the control signal is 0, the upper path is selected and vice versa. The two paths are the same except the added capacitors in the lower path. The number N denotes the amount of unit MOS capacitors used in this stage. Instead of using larger-size MOS capacitors, MOSFET's with the same size are combined in parallel to obtain better linearity. The gate area of each MOS capacitor is $4 \times 4\ \mu\text{m}^2$. According to the simulation results, at 1-V supply voltage, a unit delay of about 160 ps is obtained and the full delay range of the delay line is about 10 ns. Therefore, for an input clock of 100 MHz, the SARDLL can always exhibit a full deskew range no matter how long the two transmission lines are. Although fine unit delay can be achieved by the method, however, there still exists a disadvantage. The delay line has an offset delay about 9.2 ns (simulation result), even when the control word is 0. The larger the offset delay is, the longer the loop delay T_{loop} would be. Now, T_{delay} can be rewritten as

$$T_{\text{delay}} = T_{\text{constant}} + T_{\text{excess}} = T_{\text{constant}} + n \cdot T_{\text{unit}} \qquad (3)$$

where T_{constant}, T_{excess}, T_{unit}, and n denote the constant delay, excess delay, unit delay, and the control word of the delay line respectively.

B. Phase Comparator and Frequency Divider

The schematic as well as the operation principle of the PC is shown in Fig. 7. Two true-single-phase clocking (TSPC) D-flip-flops (DFF's) are used to sample the input clock signal. The buffers connected to *fb* and *ref* lines are the same as those used in the digital-controlled delay line and the numbers below each buffer represent the *control* signal. The clock inputs of the two sampling DFF's have a timing difference equal to the unit delay, thus forming a *lock detecting window*. When the input clock is located outside the window, the SARDLL is said unlock and *LD* is always 0 in this situation. If the feedback clock leads the input one, both outputs of the DFF's are 0, and *Comp* is 1. If the feedback clock lags the input clock, both outputs of the DFF's

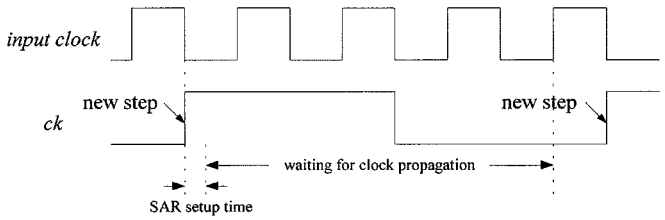


Fig. 8. SAR timing diagram.

are 1, and *Comp* turns to be 0. Once the input clock enters the window, the outputs of the DFF's will be 0 and 1, respectively, and the SARDLL is locked. Once the SARDLL is locked, *LD* will become 1 and disable the 6-bit SAR control logic immediately. The maximum static phase error between the input and output clocks will be equal to the unit delay.

Since the PC is realized with two DFF's to detect the rising edge of the input clock signal, the metastable problem is of great concern because the clock and the data input can switch exactly at the same time in this case. When the metastable problem occurs, the SARDLL can still be locked although the decision result of the PC is ambiguous. However, the static phase error between input and output clocks would be larger than one unit delay.

Theoretically, the number of bits of the control SAR determines the number of clock periods needed for the SARDLL to achieve lock. However, this is not true in practice. In some applications, after receiving a decision signal from the PC, the DLL will need time to respond before receiving the next decision result. The needed response times are the times the input clock propagates through the variable delay line $T_{\rm delay}$, the feedback transmission line $T_{\rm fb}$, and buffers. The longer the transmission line is, the more clock periods the DLL needs to respond. Otherwise, the decision results of the PC will not be faithfully reflected. This will result in DLL malfunction. Thus, a negative-edge triggered divide-by-4 frequency divider is used in the design to lower the frequency the SAR operates at. The division ratio of the frequency divider determines the maximal length of the transmission line which can be used in the deskew buffer without system malfunction. Fig. 8 shows the timing diagram. The 6-bit SAR control logic operates at one-fourth of the input clock frequency. Obviously, the side effect is an increase in lock time. The lock time of the SARDLL becomes 24 ($=6\times 4$) clock cycles for a 6-bit SAR.

C. Successive Approximation Register

The SAR in the DLL determines the value of each bit of the control word in a sequential manner, based on the output of the PC. From the chip area's point of view, it is an important building block in the SARDLL. The more compact and efficient the SAR is, the less chip area the SARDLL will occupy. The conventional SAR is composed of two sets of registers: one set stores the conversion results and the other one is used to guess the result. A nonredundant SAR, which needs only one set of registers, has been proposed and adopted in the design of the SARDLL [9].

The basic structure of the 6-bit SAR is a multiple input shift register, as shown in Fig. 9. Fig. 10 shows the internal structure of each multiple input shift register and the associated truth table. It consists of a positive-edge triggered DFF, a decoder, and a multiplexer. By adding a multiplexer and a decoder to each flip-flop, the three different inputs can be selected according to the truth table in Fig. 10. Whenever the flip-flop's are triggered, the kth flip-flop will have three different data inputs coming from: 1) the output of the $(k+1)$th flip-flop (shift right); 2) the output of the PC, *Comp* (data load); and 3) the outputs of the kth flip-flop itself (memorization).

To start the sequence, all the flip-flop's are forced to be in the initialization state (*Step 0*) with the control word equal to be 100 000, i.e., all the outputs of the flip-flops are 0 except that of the 6th flip-flop (b_5). This is accomplished by connecting the *set* input of the 6th flip-flop and the *clear* input of all the other flip-flops to a control signal, $\overline{Start}$, which sets the 6th flip-flop and clears all the other bits during the initialization state. The delay line now provides an excess delay of $0.5T_{\max}$ where $T_{\max}$ denotes the maximal excess delay provided by the delay line. The PC examines whether the feedback clock leads the input clock or not and generates the comparison result, *Comp*. Assume that *Comp* is 1, i.e., the feedback clock leads the input clock. This means that the excess delay which the delay line provides is not enough. Since all the outputs of the flip-flop's except that of the 6th one are 0, the 6th flip-flop will be in the *data load mode*. Thus, *Comp* will be loaded into the 6th flip-flop when SAR enters *Step 1*. Besides, since all of the other bits are enabled and all of their outputs are 0, they are all in the *shift right mode*. When the 6-bit SAR enters *Step 1*, the output of the $(k+1)$th flip-flop will be loaded into the kth flip-flop.

After entering *Step 1*, the control word becomes 110 000 and the delay line now provides an excess delay of $0.75T_{\max}$ since b_4 is now "guessed" as 1. Again, the PC repeats the same procedure to check the phase error between the feedback clock and the input clock. Assume that *Comp* is 0 this time, i.e., the feedback clock lags the input clock. This means that the excess delay the delay line provides is too large. Since the 5th flip-flop is in the *data load mode*, *Comp* will be loaded into the 5th flip-flop when the 6-bit SAR enters *Step 2*. This means that the "guessed" 1 will be cleared from the 5th flip-flop, otherwise the delay will be too long. Now, the 6th flip-flop is in the *memorization mode* and the rest of the flip-flops are in the *shift right mode* except the 5th flip-flop.

The SAR keeps on "guessing" a digital control word to see whether the delay is properly provided or not. To *stop* the sequence, a control signal *Stop* is applied to the OR chain. When *Stop* is active, all the flip-flops will be forced to be in the *memorization mode* immediately. The *Stop* signal is activated either when the last step of the successive approximation sequence ends or when the DLL is locked, which is indicated by one of the outputs of the PC, *LD*. At this moment, the optimal excess delay is inserted to deskew the clock signal. There exists two criteria to stop the operation of the SAR and this is different from the design of SAR analog-to-digital converters (ADC's). The additional DFF in the right end of Fig. 10 is to represent whether the DLL is locked. In the beginning, it is cleared by the $\overline{Start}$ signal to enable all the other flip-flops. After the searching sequence is finished or the SARDLL is locked, the output of this DFF will be 1 and all the other flip-flops will be disabled.

D. Initial Circuit

Finally, the SARDLL needs an initial circuit to properly reset the SAR whenever the *Start* signal is active. The initial circuit,

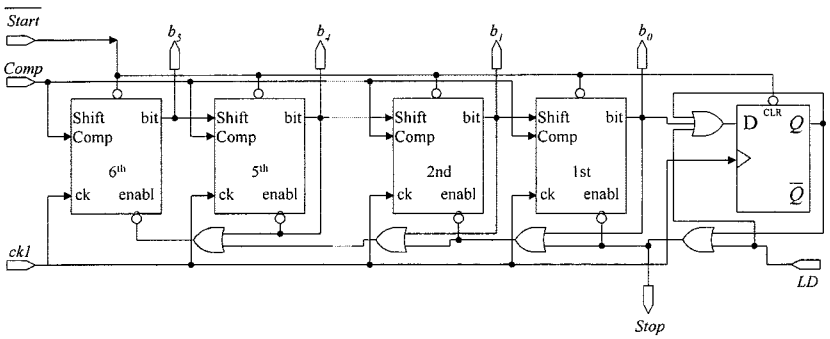


Fig. 9. 6-bit SAR circuit [9].

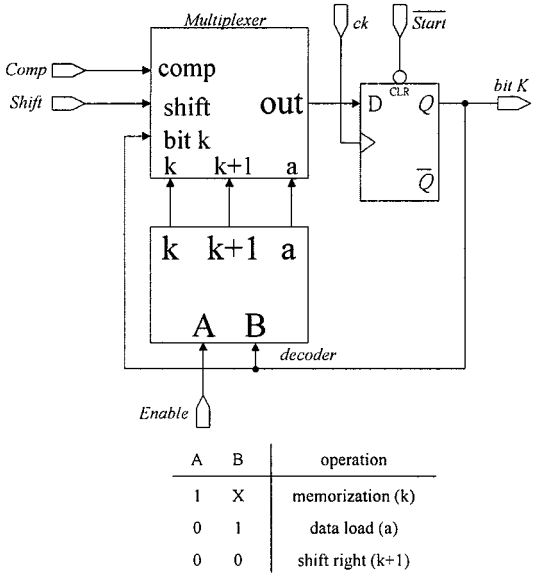


Fig. 10. Internal structure of the kth flip-flop.

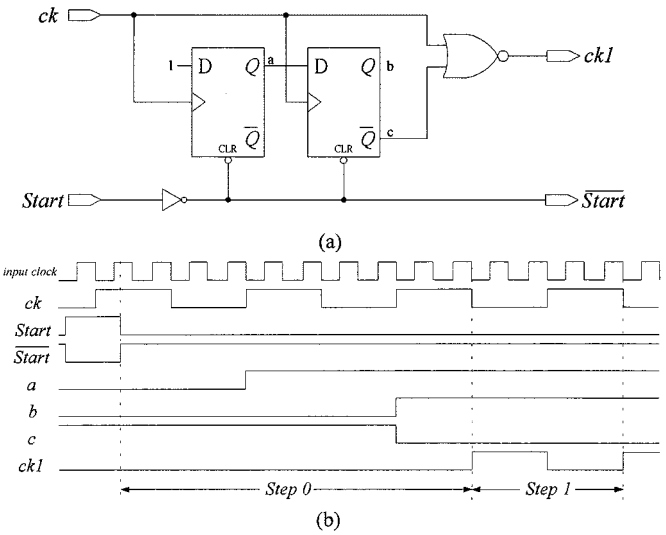


Fig. 11. (a) Schematic and (b) timing diagram of the initial circuit.

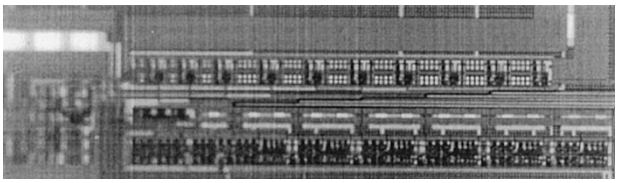


Fig. 12. Microphotograph of the SARDLL.

as well as its timing diagram, is shown in Fig. 11. The OR gate acts as a switch to enable/disable the clock signal *ck1* of the SAR. When *Start* is 1, the SAR is forced to be in the initialization state (*Step 0*), and the two DFF's in Fig. 11 are also cleared to disable *ck1*. Thus, *ck1* is forced to be 0 until node c becomes 0. The number of input clock cycles that *Step 0* needs depends on when the negative-edge of *Start* occurs, and it will not exceed 10 cycles. Therefore, the longest lock time of the SARDLL becomes 30 $(=10+5\times 4)$ input clock cycles, or 300 ns for an input clock of 100 MHz.

IV. Experimental Results

The proposed SARDLL has been fabricated in a 0.25-μm standard n-well DPTM CMOS process. Fig. 12 shows the microphotograph of the SARDLL. The active area of the chip is about $800\times 170\ \mu\text{m}^2$.

First, the digital-controlled delay line is measured at the supply voltage of 1.1 and 1 V. Fig. 13 shows both the experimental results of coarse and fine tuning sections of the delay line and the associated nonlinear characteristics. The average unit delay is 120 and 150 ps and the full excess delay time the delay line can provide is 7.48 and 9.49 ns, respectively. However, the measured curves are not very linear. It mainly results from the device mismatch of either MOS's or capacitors between the delay stages. In addition, the measured absolute delay time is much longer than the simulated result due to the extra propagation delays of the input and output buffers.

There are two kinds of transmission lines involved in the test of the SARDLL at the supply voltage of 1.1 V: on-board microstrip lines and coaxial cables. Since the characteristic impedance Z_0 of the transmission lines is 50 Ω, it is difficult to drive the line only by the on-chip output buffer under the 1.1-V supply voltage. As a result, an off-chip driver is included for test convenience.

A microstrip line with length of 45 cm is laid out on the test PCB. The propagation delay of the transmission line is about 2.76 ns. Experiments using input clocks with different frequencies, 100 and 90 MHz, are performed. Fig. 14 shows the waveforms of the two clock signals when the SARDLL is locked. All the digital signals, such as *Start*, *Stop*, and the control word are

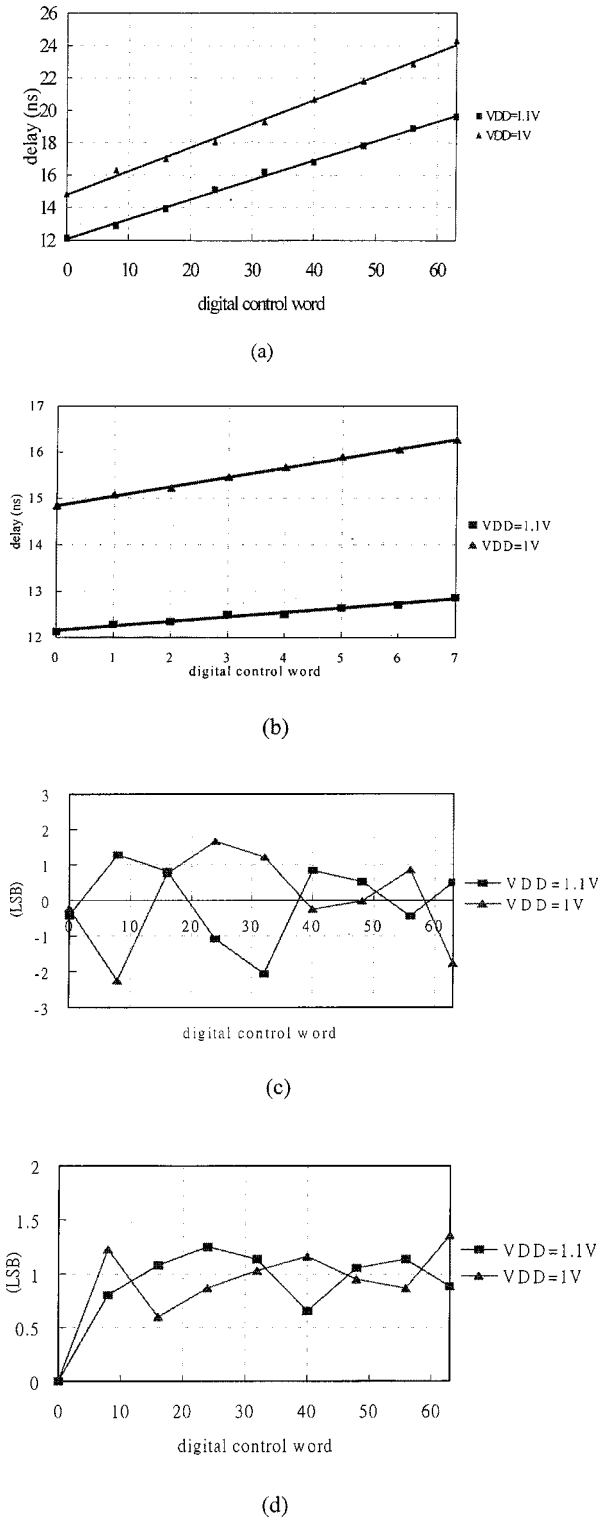


Fig. 13. Measured delay line characteristic. (a) Coarse tuning. (b) Fine tuning. (c) Integral nonlinearity (INL) error. (d) Differential nonlinearity (DNL) error.

also measured by HP16500B Logic Analysis System to show the transient lock-in process and to determine the lock time. Table I summaries the measured results, including the control word and the lock time when the SARDLL is locked. Coaxial cables with different length are also tested at the input clock frequency of 100-MHz. Fig. 15 shows the associated waveforms.

TABLE I
MEASURED RESULTS FOR 45 cm MICROSTRIP LINE.

Clock Frequency (MHz)	Locked Word	Lock Time (cycle)
100	111111	30
90	001100	18

TABLE II
MEASURED RESULTS FOR DIFFERENT COAXIAL CABLE AT 100 MHz.

Length (cm)	Locked Word	Lock time (cycle)	Skew (ps)
30	111110	22	200
60	110000	30	250

TABLE III
PERFORMANCE SUMMARY OF THE SARDLL.

Technology	0.25-μm
Supply voltage	1.1-V
Power	*3.3 mW @ 100-MHz
rms jitter	11.3 ps @ 100-MHz
Max. lock time	30 clock cycles
Chip area	800 × 170 μm^2

*not including off-chip driver's

Again, Table II summaries the measured results. Both the waveforms in Figs. 14 and 15 are sinusoidal-like. This is due to the *LC*-tank load of the off-chip driver.

According to the measured results, the SARDLL can always achieve lock within 30 clock cycles. However, the measured skew between the two input signals of the PC is somewhat larger than the unit delay. One possible reason is that the minimum detectable phase error for the PC is larger than the unit delay. Another reason would be the nonlinearity of the delay line. The metastable phenomena of the PC also accounts for the larger static phase error. All of the nonideal features mentioned above would result in a control word which is different from the desired one when the SARDLL is locked. If a PC with higher resolution and a delay line with better linearity are available, a SARDLL clock-deskew buffer with high performance can be realized. Moreover, if the SARDLL can respond to the decision results of the PC quickly, the division ratio of the frequency divider can be reduced, i.e., the 6-bit SAR control logic can operate at higher clock frequencies. This will also help to shorten the lock time.

The measured rms and peak-to-peak jitters of the 100-MHz output signal of the SARDLL are 11.3 and 95 ps, respectively, as shown in Fig. 16. The measured power consumption is 3.3 mW including the input and output buffers (not including off-chip drivers) when the SARDLL is locked at 100-MHz clock input. Table III summaries the performance of the proposed SARDLL.

V. CONCLUSION

In this paper, a low-voltage clock-deskew buffer fabricated in a 0.25-μm standard CMOS technology is presented. The clock-deskew buffer utilizes a SARDLL technique. This architecture uses a binary-search method to quickly find an optimal delay time between the input and output clocks for clock synchronization, and achieves short lock time, compared to the conventional

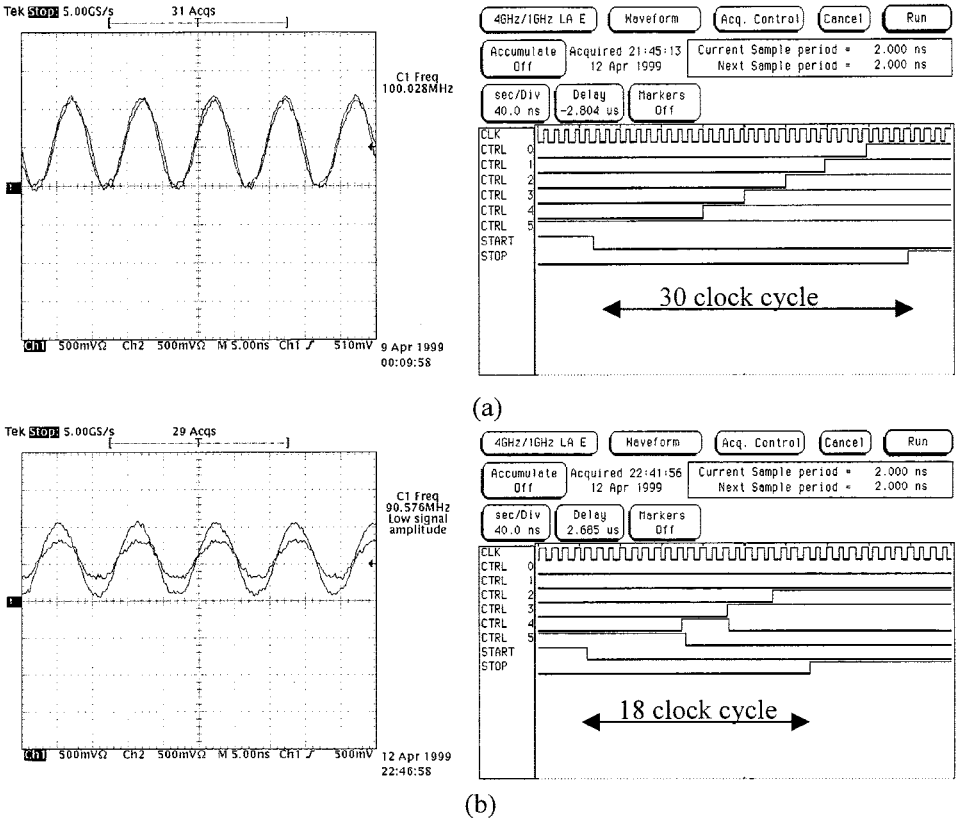


Fig. 14. Test results with microstrip lines. (a) 100-MHz clock. (b) 90-MHz clock.

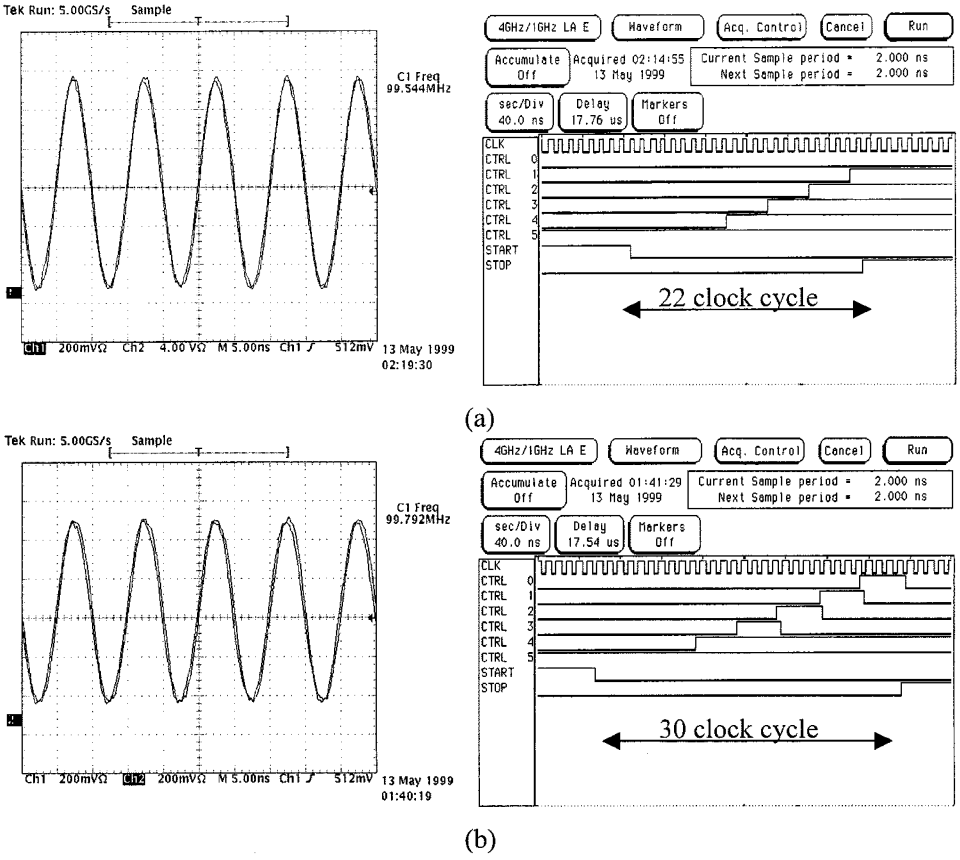


Fig. 15. Test results with coaxial cables. (a) 30 cm cable. (b) 60 cm cable.

RDLL or CDLL. When the input clock frequency is 100 MHz, the measured output clock rms and peak-to-peak jitter are 11.3 and 95 ps, respectively. The chip dissipates a power of 3.3 mW (not including off-chip driver's) at 1.1-V supply voltage. The

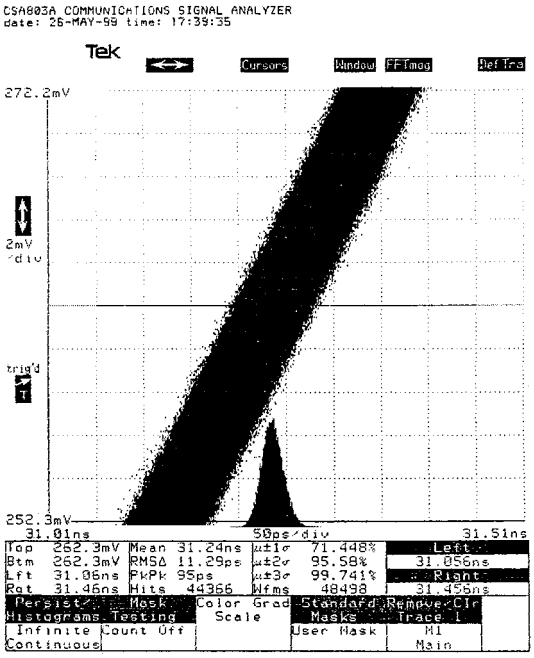


Fig. 16. Measured clock jitter.

measured lock time is within 30 input clock cycles, and the proposed SARDLL is very suitable for high-speed low-voltage low-power clock-deskew applications.

ACKNOWLEDGMENT

The authors would like to thank SHARP Company, Japan, for the fabrication of the chip.

REFERENCES

[1] Y. Okajima, M. Taguchi, M. Yanagawa, K. Nishimura, and O. Hamada, "Digital delay locked loop and design technique for high-speed synchronous interface," *IEICE Trans. Electron.*, vol. E79-C, no. 6, pp. 798–807, June 1996.

[2] H. Sutoh and K. Yamakoshi, "A clock distribution technique with an automatic skew compensation circuit," *IEICE Trans. Electron.*, vol. E81-C, no. 2, pp. 277–283, Feb. 1998.

[3] S. I. Liu, J. H. Lee, and H. W. Tsao, "Low-power clock-deskew buffer for high-speed digital circuits," *IEEE J. Solid-State Circuits*, vol. 34, pp. 554–558, Apr. 1999.

[4] A. Hatakeyama, H. Mochizuki, T. Aikawa, M. Takita, Y. Ishii, H. Tsuboi, S.-Y. Fujioka, S. Yamaguchi, M. Koga, Y. Serizawa, K. Nishimura, K. Kawabata, Y. Okajima, M. Kawano, H. Kojima, K. Mizutani, T. Anezaki, M. Hasegawa, and M. Taguchi, "A 256-Mb SDRAM using a register-controlled digital DLL," *IEEE J. Solid-State Circuits*, vol. 32, pp. 1728–1734, Nov. 1997.

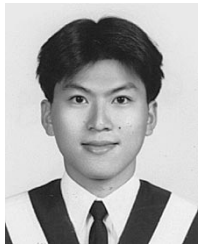
[5] F. Lin, J. Miller, A. Schoenfeld, M. Ma, and R. J. Baker, "A register-controlled symmetrical DLL for double-data-rate DRAM," *IEEE J. Solid-State Circuits*, vol. 34, pp. 565–568, Apr. 1999.

[6] H. Sutoh, K. Yamakoshi, and M. Ino, "Circuit technique for skew-free clock distribution," in *IEEE Custom Integrated Circuits Conf.*, 1995, pp. 163–166.

[7] T. Saeki, Y. Nakaoka, M. Fujita, A. Tanaka, K. Nagata, K. Sakakibara, T. Matano, Y. Hoshino, K. Miyano, S. Isa, S. Nakazawa, E. Kakehashi, J. M. Drynan, M. Komuro, T. Fukase, H. Iwasaki, M. Takenaka, J. Sekine, M. Igeta, N. Nakanishi, T. Itani, K. Yoshida, H. Yoshino, S. Hashimoto, T. Yoshii, M. Ichinose, T. Imura, M. Uziie, S. Kikuchi, K. Koyama, Y. Fukuzo, and T. Okuda, "A 2.5-ns clock access, 250-MHz, 256-Mb SDRAM with synchronous mirror delay," *IEEE J. Solid-State Circuits*, vol. 31, pp. 1656–1668, Nov. 1996.

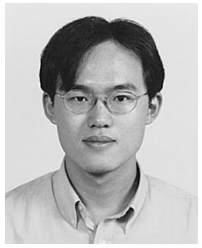
[8] U. Tietze and Ch. Schenk, *Electronic Circuits—Design and Applications*. New York, NY: Springer-Verlag, 1992, p. 703.

[9] A. Rossi and G. Fucilli, "Nonredundant successive approximation register for A/D converters," *Electron. Lett.*, vol. 32, no. 12, pp. 1055–1057, June 1996.



Guang-Kaai Dehng (S'98) was born in Changhua, Taiwan, R.O.C., on December 9, 1973. He received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, in 1996, where he currently is working toward the Ph.D. degree in electrical engineering.

His research interests include PLL, DLL, and high-speed CMOS data transmission circuits for Gbit communication.

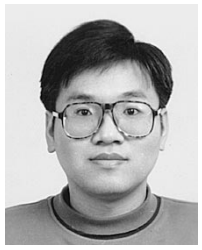


June-Ming Hsu was born in Taipei, Taiwan, R.O.C., on December 6, 1974. He received the B.S. and M.S. degrees in electrical engineering from National Taiwan University, Taipei, in 1997 and 1999, respectively.

After graduating, he joined Electronics Research & Service Organization, Industrial Technology Research Institute, Hsinchu, Taiwan, where he is currently working on radio-frequency integrated circuits for GSM/PCN transceivers. His previous work includes CMOS PLL frequency synthesizer and DLL clock deskew buffer.

Ching-Yuan Yang (S'97) was born in Miaoli, Taiwan, R.O.C., on August 17, 1967. He received the B.S. degree in electrical engineering from the Tatung Institute of Technology, Taipei, Taiwan, in 1990, and the M.S. degree in electrical engineering from National Taiwan University, Taipei, in 1996, where he currently is working toward the Ph.D. degree in electrical engineering.

His research interests are in the area of integrated circuits and systems for high-speed interfaces and wireless communications.



Shen-Iuan Liu (S'88–M'93) was born in Keelung, Taiwan, R.O.C., on April 4, 1965. He received the B.S. and Ph.D. degrees in electrical engineering from National Taiwan University, Taipei, Taiwan, R.O.C., in 1987 and 1991, respectively.

During 1991–1993 he served as a Second Lieutenant in the Chinese Air Force. During 1991–1994, he was an Associate Professor in the Department of Electronic Engineering, National Taiwan Institute of Technology, Taiwan. He joined the Department of Electrical Engineering, National Taiwan University, in 1994 and he has been a Professor since 1998. He holds eight U.S. patents and thirteen R.O.C. patents, some pending. His research interests are in analog and digital integrated circuits and systems.