A CMOS Oversampling Data Recovery Circuit With the Vernier Delay Generation Technique

Jun-Young Park* Jin-Ku Kang* *Regular Members*

ABSTRACT

This paper describes a CMOS data recovery circuit using oversampling technique. Digital oversampling is done using a delay locked loop circuit locked to multiple clock periods. The delay locked loop circuit generates the vernier delay resolution less than the gate delay of the delay chain. The transition and non-transition counting algorithm from 4x oversampling was implemented for data recovery and verified through FPGA. The chip has been fabricated with 0.6um CMOS technology and measured results are presented.

Ⅰ. Introduction

Data recovery systems using the data-oversampling method have been developed in CMOS technology [1][2][3]. In their oversampling technique, the clock (or data) is propagated through delay taps for generating multi-phase clocks (or data), limit the sampling resolution to the minimal intrinsic gate delay. Finer delay requires the use of faster, more expensive technologies. In order to generate precise delays with significantly finer resolution than an intrinsic gate delay without the use of state-of-the-art integrated circuit technology, some design approaches have been presented [4][5]. These approaches though have a complex architecture to implement.

This paper uses a single delay chain that is locked to multiple clock periods to achieve a delay resolution less than one delay cell delay time. The sampling resolution can be smaller than with conventional DLL. This finer sampling resolution provides the capability of oversampling higher data rate with a high oversampling ratio. Consequently, jitter tolerance can also be improved [6].

The multi phase signals from delay chain are used for oversampling input data. Proper data boundary position is decided from sampled data bit array to perform the correct data recovery.

In the data recovery circuit each bit of input data is oversampled. A proper sample among the oversampled points per 1bit input data is then selected as recovered data by locating the NRZ input data boundary.

Section 2 describes the overall architecture of the implemented data recovery circuit and section 3 discusses the basic theory of the components. Circuit design is presented in section 4 and experimental results are given in section 5, followed by the conclusions.

Ⅱ. Architecture

The architecture of the data recovery circuit is shown in Fig. 1. The DLL for a multi-phase clock generator is locked by an external clock signal that runs at one-eighth the data rate for the 1:8 demultiplexing data recovery. The delay stages of the delay chain are tapped to produce the sampling clocks and these clocks are rearranged in time sequence. A bank of input samplers uses these clock edges to oversample the input data. Transitions in the input data bit stream are detected and processed to decide proper input data

* 인하대학교 전자·전기컴퓨터공학부
논문번호 : J99011-1208, 접수일자 : 1999년 12월 8일

bit boundary and this information is used to pick the oversampled points to recover data. To process the oversampled data in parallel, first in first out(FIFO) registers are needed. The sampled points are held at the register by the time needed to decide data boundary position so that correct bits are selected. In our architecture, 4X oversampling is chosen. This means the sampler oversamples four times per bit. Since eight input data are processed in one clock cycle, 32 clock edges and latches in the samplers are required. It is a very practical assumption that in the worst case, among the four samples, two sampled points closer to the each data boundaries are uncertain due to jitter. However, the other two sampled points inside boundaries would be unaffected from noise and these two sample points play an important role to robust data recovery, which is explained later.

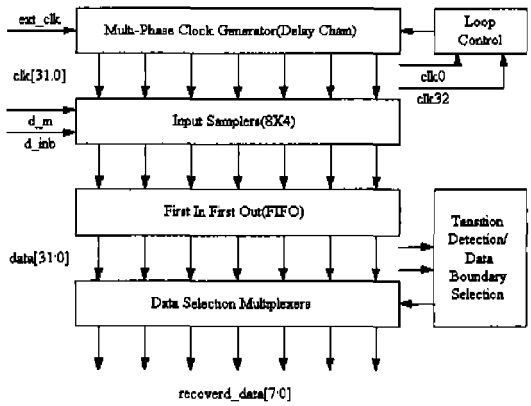


Fig. 1 Architectural Floor Plan

Ⅲ. Multi-Phase Clock Generator

The general method that makes signals equally spaced in time domain is to tap a chain of delay elements, which the delay time is controlled by a delay locked loop(DLL). When the two input signals to phase detector from delay chain are in phase, *i.e.* the DLL is locked, the following equation is satisfied between delay time per unit delay cell(D), external clock period(T), and the number of total delay cell stages(N).

$$D \cdot N = n \cdot T \quad (1)$$

,where n is an integer number. In conventional DLL's, the delay chain is locked to the single clock period(when n=1). From the above equation, the delay time D can be reduced by increasing the total stage number N but can not be less than intrinsic gate delay of the each delay stage. Now we lift the constraint that the delay chain should be locked to the one clock period(n=1). Here, the number of locking clock periods, n, can be any integer. Table 1 shows the delay time ($D = nT/N$) for various stage numbers (N) and the numbers of clock periods(n) under the fixed external clock period, T = 8ns.

The bolded numbers in Table 1 are the cases when the ratio $N/n=T/D$ makes irrational number and the other numbers are the case when the ratio makes rational number. The combinations of the number of delay stages(N) and the number of clock periods to DLL(n) which generate the bold styled delay values are the conditions for producing a delay resolution less than the delay of single delay cell.

Fig. 2 and Table 2 show the case when the total number of delay stage (N) is 32, the number (n) is 3, and the external clock period (T) is 8ns. These setting displays that each delay cell generates 750ps delay(between 0th and 1st stage, between 1st and 2nd,...), but actually the signals have 250ps difference in time sequence(e.g. between 0th and 11th, between 11th and 22nd,...). The numbers in 'N' column in Table 2 stand for the Nth delay stage in the delay chain.

As shown in Fig.2, the signal from 11th delay stage has the delay difference of 250ps with the signal going into 1st delay stage. This is because the actual delay is the modulus of the clock period on the total delay time after 11th stage. The effective delay, δ_i at ith delay stage is therefore can be written as:

$$\delta_i = (i \times D) \bmod T \quad (3)$$

Thus the circuit can oversample 4 times per

1Gb/s NRZ data.

Table 1. The delay time(D) per single stage for various stage numbers(N) and the number of clock periods(n). (unit : ps, T=8ns)

N \ n	1	2	3	4	5	6
24 stage	333.3	666.6	1000	1333	1666	2000
32 stage	250	500		1000	1250	
40 stage	200	400		800	1000	
48 stage	166.6	333.3	500	666.6	833.3	1000
56 stage	143.8	285.7		571.4	714.2	
64 stage	125	250		500	625	

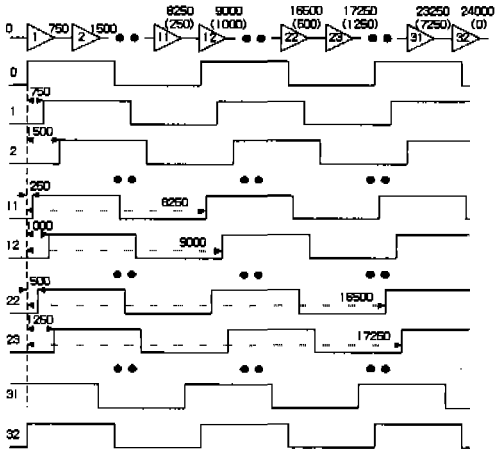


Fig. 2 Waveform of delay stage in case of N=32, n=3, and T=8ns. The delay difference between 0th stage and 11th stage is 250ps under the absolute gate delay of 750ps(N/n and T/D is an irrational number in this case.)

3.2 Data Recovery Algorithm

Fig. 3 shows a simplified data recovery process. Data recovery process requires finding and tracking the bit boundaries from sampled points. Then the proper sample is chosen as the recovered data. At the bit boundary, the signal has transition either from zero to one or one to zero. Thus finding transition points lead to the bit boundary. And the correct bit boundary gives the correct recovered data. Thus the finding the correct bit boundary is the most important step for the data recovery.

The decision logic first detects data transition points by an XOR of adjacent samples. The bit

Table 2. The time delay of delay stage with respect to 0th stage in the order of time sequence.(N=32, n=3, and T=8ns)

N	D (ps)	N	D (ps)	N	D (ps)	N	D (ps)
0		24	(18000)	16	(12000)	8	
11	(8250)	3		27	(20250)	19	(22500)
22	(16500)	14	(10500)	6		30	(22500)
1		25	(18750)	17	(12750)	9	
12	(9000)	4		28	(21000)	20	(15000)
23	(17250)	15	(11250)	7		31	(23250)
2		26	(19500)	18	(13500)	10	
13	(9750)	5		29	(21750)	21	(15750)

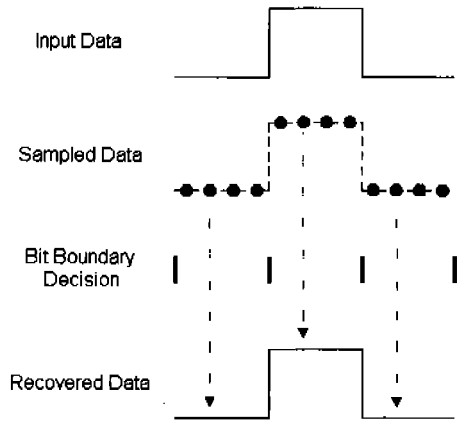


Fig. 3 Data recovery process

boundary can be deduced from transitions. If the data boundaries are determined, samples in two points after each the boundary position are selected as the recovered data. Since the single bit input data is oversampled 4 times, the bit boundary reappears in every four oversamples in 32 oversamples acquired from 8 bit input data. In order to determine the bit boundary, the 32 samples can be grouped into four.

$$x_l[n] = x[4m + l] \quad (0 \le l \le 3) \qquad (4)$$

where n is the sample position among the 32 stages, m denotes the mth bit among 8 input data oversampled in one clock cycle($0 \le m \le 7$), and l is the recalculated sample position with respect to a single data bit. We denote position 'a' when

the transition occurs between $x_0[n]$ and $x_3[n]$, position 'b' between $x_3[n]$ and $x_2[n]$, position 'c' between $x_2[n]$ and $x_1[n]$ and position 'd' between $x_1[n]$ and $x_0[n]$.

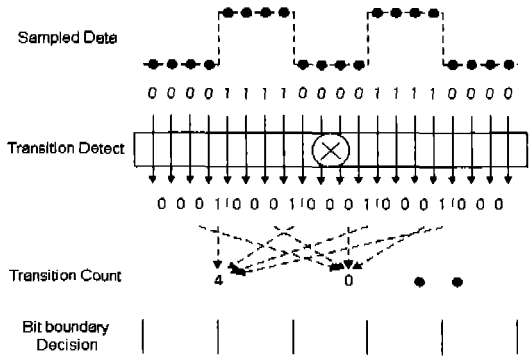


Fig. 4 Bit boundary decision with the transition count of sampled data

Fig.4 shows an example of the boundary detection process with a portion of oversamples. In order to find the bit boundary, transitions occurring at the same recalculated position are counted. The sample position with the largest total is the bit boundary. Fig. 4 illustrates the maximum count of the position is at 'a' and thus the bit boundaries are between 4th and 5th sample, 8th and 9th, 12th and 13th.... The decision logic updates the bit boundary in every clock cycle equivalent to 8 bit input data period. If there is no data transition during 8 consecutive input data, the transition position 'a' is chosen for the bit boundary.

If using the above transition counting algorithm only, two positions can be possible candidates for the bit boundary illustrated in Fig. 5 under some noisy conditions. If the samples from jittered input data has 5 or 3 sampling points per single data bit instead of the nominal 4 sampling points, the transition count calculation gives two positions('a' and 'b') as the bit boundary candidates. If the position 'a' is chosen as the boundary, the correct data recovery is performed. But, if the position 'b' is selected as the bit boundary, the wrong data would be recovered. To resolve this problem, we also look for the position which has no transition, which is the position 'c' in Fig. 5. This no-transition position 'c' indicates that two adjacent sampled points, $x_2[n]$ and $x_1[n]$, should be placed in the middle of one bit sampling window. Therefore, the transition position 'b' can not be the boundary in the case shown in Fig. 5. This can be derived from the assumption that the maximum possible jitter hurts only two sampling points near bit boundary. This non-transition counting algorithm avoids the possible error in the transition count algorithm only. And this algorithm can be implemented in the 4X oversampling technique. Unlike other techniques [7][8], this transition and non-transition counting algorithm can also be applied to the data communication for data recovery without any preamble period.

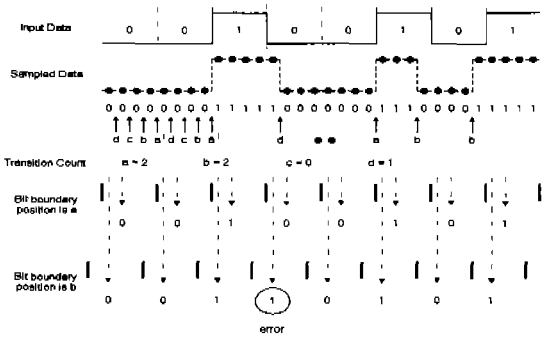


Fig. 5 The information from transition count can give an incorrect bit boundary position with only the transition count algorithm under jittered condition. The non-transition count('c') algorithm eliminates incorrect possible bit boundary('b').

Ⅳ. Circuit Implementation

A block diagram of a DLL for generating a fine resolution multi-phase clock is shown in Fig. 6. It consists of a phase detector, a charge pump, and a delay chain of 32 stage. The transistors in delay cell are locked at 750ps delay time. The real sampling resolution is 250ps if the signals are rearranged in time. Since the delay chain should be locked at only three clock period time,

there needs some measures to prevent false locking due to process or operating condition variations. We placed an additional delay chain composed by ten delay stage(750ps) plus a single delay stage(500ps), which is locked at 8ns clock and assisting to maintain the delay value as 750ps. Also, 3 extra pins are assigned to control the delay cell locked at 750ps externally.

The D flip-flop in the phase detector is a single phase clock D-F/F with a clear function [9]. The phase detector using flip-flop is not sensitive to differences of the duty-cycle between the inputs and D flip-flop of only 11 transistors is used to achieve dead zone free. The circuit of the input sampler is based on the reference [7]. Input data and the sampling clocks are provided in differential manner. The sampling ends on the falling edge of the clock, holding the most recent sampled data that arrived for regenerative amplification for half cycle.

Due to the presence of multiple clocks in the sampler, the sampled points could be overwritten before proceeding to the data processing block. To avoid this, the half-segment register deskewing with a FIFO structure is used [10]. In order to latch the oversampled bits in parallel, they must first be aligned in time (de-skewed). Instead of implementing area-consuming delay units on de-skewing lines, half-segment latch stages were used. This allows wider setup/hold time windows for the parallel latching. FIFOs are used for shifting samples without being overwritten by the next incoming sampling clock.

The transition detection and boundary decision block is shown in Fig.7. As described in the data recovery algorithm, both the transition and non-transition count are used for the boundary position decision.

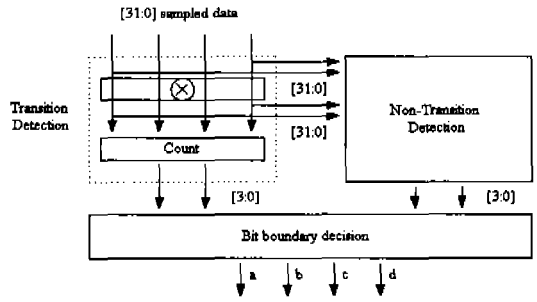


Fig. 7 Block diagram of the boundary decision circuit

V. Simulated and Measured Results

The logic simulation of the data recovery algorithm has been performed by Altera's Flex 10K FPGA. Fig 8 shows test vectors including possible samples from noisy conditions. It shows that marked data are to be recovered and data boundary(window) position are illustrated using a bar and identified. Fig. 9 shows that the proper data boundary(window) is determined. As the clock proceeds, the bit boundary is A, A, C, C, C, B , A, A,..., which is the same as predetermined values in Fig. 8. In Fig. 10 the recovered data are shown. The recovered data sequence is the same as those in Fig. 8.

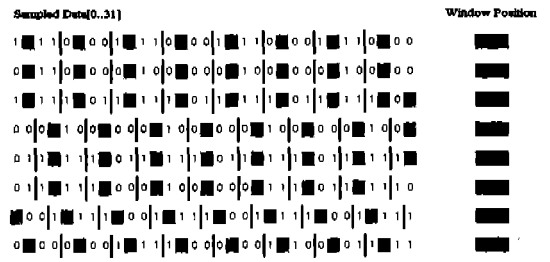


Fig. 8 8 test vectors to the digital data recovery logic(The data transition position and data to be recovered are marked in gray)

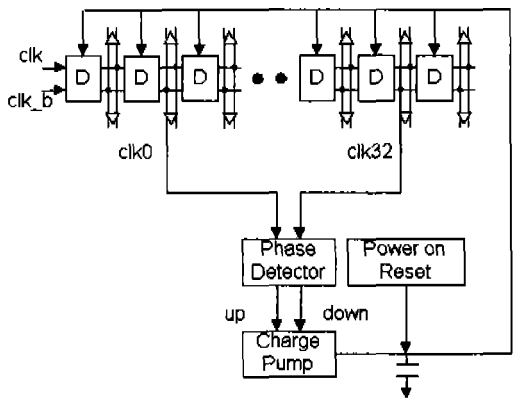


Fig. 6 Block diagram of the designed DLL

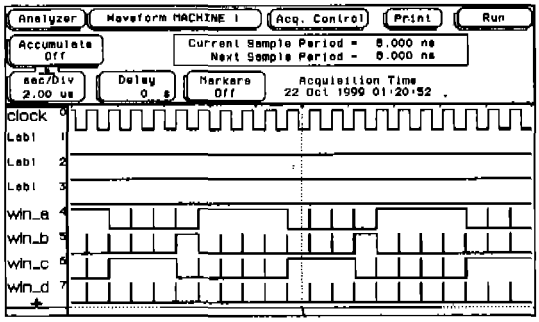


Fig. 9 Clock signal(top plot) and data boundary position (from 5th to 8th plot : win_a ~ win_d)

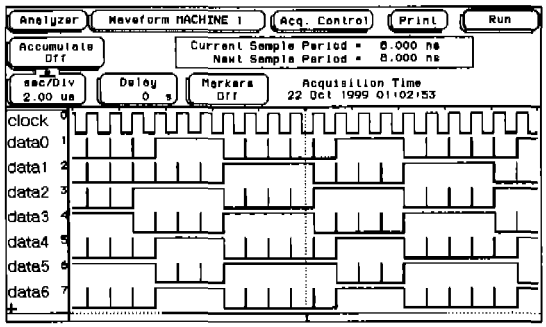


Fig. 10 Clock signal(top plot) and recovered data(from 2nd to 8th plot : data0 ~ data6)

The whole circuit is implemented in a standard 0.6um CMOS technology and the chip has been fabricated. The die photo is shown in Fig. 11. Post layout simulation has been done using HSPICE. The design and performance statistics are summarized in Table 3.

The external clock signal to the delay chain is 1V swing(3.8V - 4.8V) at 125MHz and the DLL is locked to the three clock period time(3×8ns). The differential input data is a 1 Gb/s with 0.5V swing (1.25V - 1.75V).

Table 3. Design and Performance Results

Technology	0.6um CMOS
Transistors	14700
Die size(core)	3.1mm X 1.9mm
I/O	80pin QFP
Power supply	5 volt
Input Data Rate	1Gb/s
Clock frequency	125MHz
Clock jitter	100ps
Output Data Rate	125Mbps
Sampling resolution	250ps
Power consumption	800mW

The measured output signals from five channels among eight output channels are shown in Fig.12. For example, the measured output signal of channel 2 is '1010....' pattern. The pulse width of '1' and '0' is 8ns(125Mb/s). The other channels patterns are '0001..', '1100..' and '1110..' Various input patterns were supplied for checking the correct data recovery. Due to the demultiplexed structure and the lack of BER test equipment, the BER test using PRBS pattern could not be performed yet.

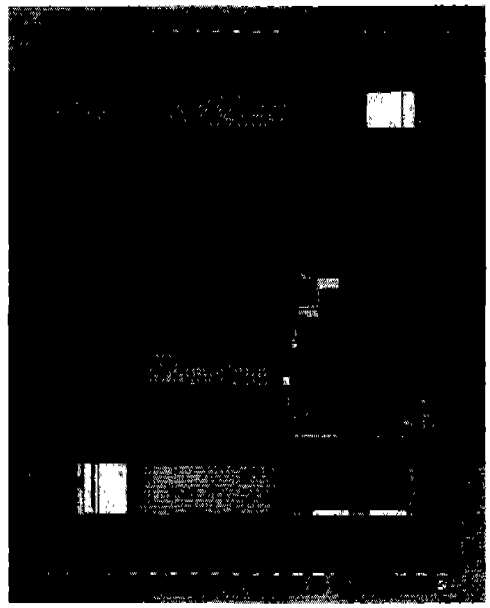


Fig. 11 Chip Microphotograph

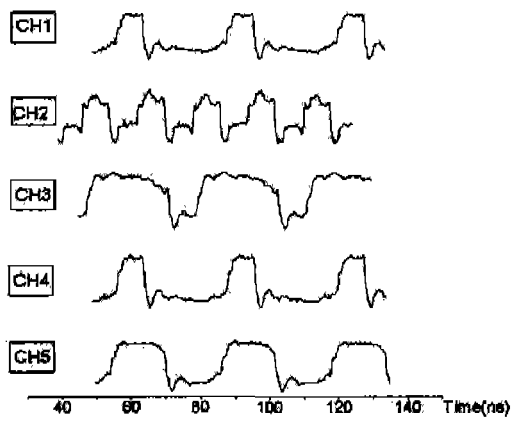


Fig. 12 Measured output signals

Instead, the output for various known input data patterns with maximum bit length of 16 bits were checked and compared. In all input data patterns, data were recovered without any error. The sampling clock jitter was measured by checking the output while sweeping a input transition. The time period producing an unstable output is measured as clock jitter which was about 100ps.

Ⅵ. Conclusion

We presented a 1 Gb/s CMOS data recovery circuit using 4X oversampling technique. To achieve a fine sampling resolution, a single delay chain locked to multiple clock periods is proposed. This approach can realize the delay resolution less than one gate delay in the delay chain. For a robust data recovery, the transition and non-transition count algorithm was implemented. The data recovery algorithm was verified through FPGA. The whole chip has been fabricated in 0.6um CMOS technology and measured.

Acknowledgement

This work was supported by the development program for the exemplary schools in information and communication from the Ministry of Information and Communication.

References

[1] J. Sonntag and R. Leonowich, "A monolithic CMOS 10MHz DPLL for Burst-Mode Data Retiming," IEEE International Solid-State Circuit Conference, pp. 194-195, Feb. 1990.

[2] M. Bazes and R. Ashuri, "A novel CMOS Digital Clock and Data Decoder," IEEE Journal of Solid State Circuits, pp 1934-1940, Dec. 1992.

[3] B. Kim, D. N. Helman, and P. R. Gray, "A 30-MHz Hybrid Analog/Digital Clock Recovery Circuit in 2um CMOS," IEEE Journal of Solid State Circuits, pp 1385-1394, Dec. 1990.

[4] S. Sidiropolous and M. Horowitz, "A Semi-Digital DLL with Unlimited Phase Shift Capability and 0.08-400MHz Operating Range," IEEE Journal of Solid State Circuits, Vol. 32, No. 11, pp 1683-1692, Nov. 1997.

[5] J. G. Maneatis, "Low-Jitter and Process-Independent DLL and PLL Based on Self-Biased Technique," IEEE ISSCC, pp 130-131, 1996.

[6] J. Kang, "Performance Analysis of Over-sampling Data Recovery Circuit," IEICE Transactions on Fundamentals of Electronics, Communications and Computer Science, Vol. E82-A, No. 6, pp 958-964, June 1999.

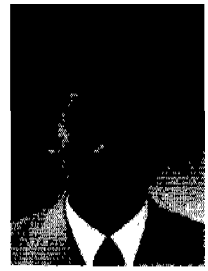
[7] C. Yang, R. Farjad-Rad, and M. Horowitz, "A 0.5um CMOS 4.0Gbps Serial Link Transceiver with Data Recovery using Oversampling," IEEE Journal of Solid State Circuits, pp 713-721, 1998.

[8] T. Fong, M. Cerisola, R. Hofmeister, and L. Kazovsky, "Ultra fast clock recovery for optical packet switched networks," Electronic Letters, pp 1687-1688, Sep. 1995.

[9] R. Rogenmoser, "1.16GHz Dual-Modulus 1.2um CMOS Prescaler," CICC, pp 27.6.1-27.6.4, 1993.

[10] J. Kang, W. Liu, and R. K. Cavin, "A CMOS High Speed Data Recovery Circuit using Matched Delay Sampling Technique," IEEE Journal of Solid State Circuits, pp 1588-1596, Oct. 1997.

박 준 영(Jun-Young Park)

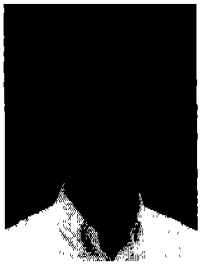


1998년 : 인하대학교 전자 · 전기 · 컴퓨터 공학부 졸업

현재 : 인하대학교 전자 · 전기 · 컴퓨터공학부 석사졸업.

<주관심 분야> CMOS고속회로 설계, VLSI, 신호처리 회로설계

강 진 구(Jin-Ku Kang)



1983년 : 서울대학교 공학사.
1990년 : New Jersey Institute of Technology 전기공학석사
1996년 : North Carolina State University, 전기 및 컴퓨터공학 박사.

1983년~1988년 : 삼성반도체,
1996년~1997년 : 미국 INTEL 선임설계연구원.
1997년 3월~현재 : 인하대학교 전자전기 및 컴퓨터 공학부 조교수
<주관심 분야> 고속 CMOS회로설계. 혼합모드 회로설계, 통신용회로설계