An 128-phase PLL using interpolation technique

Hayun Chung, Deog-kyoon Jeong , and Wonchan Kim

Abstract— **This paper presents an 125MHz, 128-phase phase-locked loop using interpolation technique for digital timing recovery. To reduce the power consumption and chip area, phase interpolation was performed over only selected windows, instead of overall period. Four clocks were used for phase interpolation to avoid the output jitter increase due to the interpolation clock (clock used for phase interpolation) switching. Also, the output clock was fed back to finite-state machine (FSM) where the multiplexer selection signals are generated to eliminate the possible output glitches. The PLL implemented in a 0.25μm CMOS process and dissipates 80mW at 2.5V supply and occupies 0.84mm^2.**

Index Terms— **phase-locked loop, phase interpolation, output jitter increase problem, glitch, clock feedback structure**

I. INTRODUCTION

Multi-phase phase-locked loops (PLL's) are usually used for digital timing recovery in order to convert the digitized timing information, the result of digital timing recovery, into phase. In the digital-to-phase conversion, the discontinuity of the digital code makes the PLL generate clock with discrete phases. Since the discrete phases do not cover every phase within the clock period, a quantization error is inevitable in multi-phase PLL. Thus, to minimize this quantization error the PLL must improve its phase resolution as high as possible.

Multi-phase PLLs are generally realized with multi-stage voltage controlled oscillator (VCO), by extracting clocks from each stage. Since each stage of the multi-stage VCO is composed of delay cells, the operation speed of the PLL can be restricted as the number of stages is increased to improve phase resolution. Usually, to achieve high phase resolution without restricting the PLL's operation speed, phase interpolation techniques are adopted.

Among the interpolation schemes, schemes that interpolate phases over only selected windows were considered in this paper rather than the overall period phase interpolation scheme to improve power and area efficiency. Newly proposed selective phase interpolation scheme using four interpolation clocks (PI4C) eliminates the jitter increase problem, which can appear when only two clocks are used for phase interpolation.

Also, to eliminate possible output glitches, an architecture that feeds output clock back to the finite-state machine (FSM) was suggested. The PLL was implemented to support 4-channel communications and fabricated in 0.25μm CMOS process. The 4-channel PLL consumes 80mW power under 2.5V supply and occupies 0.84mm^2 chip area.

II. PHASE INTERPOLATION SCHEMES

The PLL achieved 128-phase resolution to reduce the quantization error due to the discrete phase intervals. To increase phase resolution without restricting the PLL's operation speed, an interpolation technique is applied. In implementation, a 16-stage differential VCO and a 4x phase interpolator was employed – the 16-stage differential VCO generates 32 clocks with regular phase

Manuscript received November 5, 2003; revised November 26, 2003.

School of Electrical Engineering and Computer Science, Seoul National University
Shinlim-Dong, Gwanak-Gu Seoul 151-742, KOREA
TEL : +82-2-880-1306, FAX:+82-2-871-3985
Email : hayun@iclab.snu.ac.kr, dkjeong@ee.snu.ac.kr, wkim@iclab.snu.ac.kr

intervals, and then a 4x phase interpolator multiplies the number of phases by 4. Among the interpolated clocks one is selected by a multiplexer (MUX) as an output. Note that the PLL shifts its output clock phase one step left or right when the UP or DOWN signal is received from digital signal processing (DSP), respectively. Three possible phase interpolation schemes will be discussed here: one overall period phase interpolation scheme and two selective phase interpolation schemes using two clocks and four clocks.

Fig. 1 shows the operation of overall period phase interpolation scheme. In this scheme, PLL generates all the 128 equally spaced clocks and select one among them as an output clock.

This scheme provides reliable operations. It always presents 128 stabilized interpolated waveforms after the phase interpolation is once settled. Since the output clock is selected among the 128 pre-settled clocks, the output clock always appears to be stabilized. Also, the selection scheme is quite simple and straightforward. There is no complex and problematic operation like interpolation clock switching, which is needed in selective phase interpolation schemes. The only operation needed is selecting one output clock out of the 128 clocks. However, the implementation of structures for 128-clock generation and output clock selection out of the 128 clocks result in high power consumption and large chip area and degrade the PLL's efficiency. Thus, for better performance in power consumption and chip area, selective phase interpolation schemes, which interpolate phases over only selected windows, are highly recommended.

Fig. 2 shows one of the selective phase interpolation schemes – phase interpolation scheme using two interpolation clocks (PI2C). This scheme uses only two adjacent clocks among the 32 clocks generated from 16-stage differential VCO in phase interpolation [1]. The output clock is selected among the five clocks, the result of the phase interpolation between two clocks.

This scheme uses minimum number of interpolation clocks, only two clocks, for phase interpolation. Because this scheme generates only five clocks through interpolation, it presents one of the most power and area efficient performances. However, since the interpolated clocks do not cover the overall period, as the output phase moves due to the UP/DOWN signal incomings from DSP, the interpolation window should also be shifted. This increases the complexity of control block and degrades the stability of output clock waveform due to the re-interpolation. Fig. 3 shows the output clock phase shifting operation, which involves change of interpolation window.

When the successive UP (or DOWN) signals from DSP requires the output clock to be selected from the outside of present interpolation window, one of the

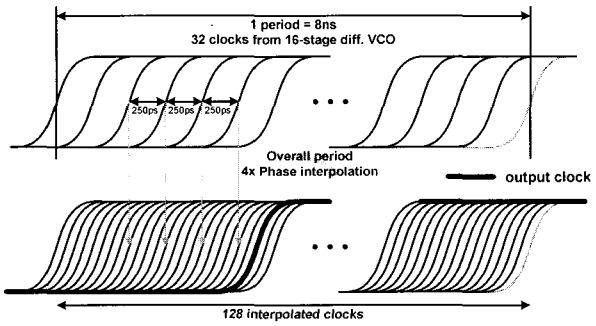


Fig. 1. Overall period phase interpolation scheme

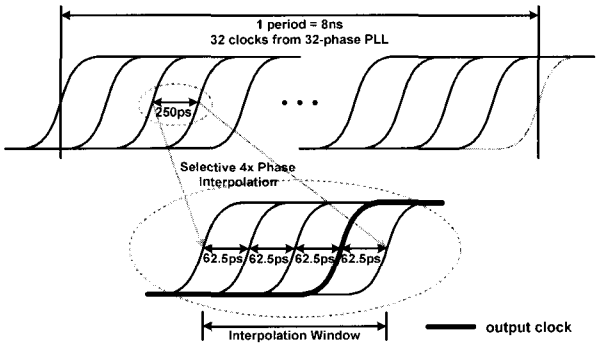


Fig. 2. Phase interpolation scheme using two interpolation clocks

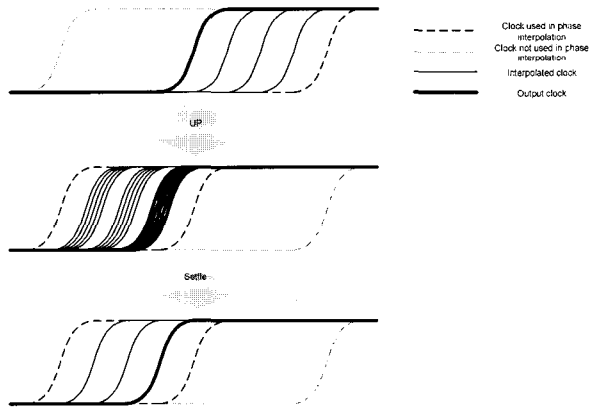


Fig. 3. Shift of Interpolation window in PI2C(Output jitter increase due to the interpolation clock switching)

interpolation clocks should be changed to shift the interpolation window left (or right). At this moment, the change of interpolation clock and the change of output clock selection occur simultaneously. After the interpolation clock switching, the phases should be re-interpolated and it takes about 100ns to settle down the resulted waveforms. This means, an unsettled waveform appears in the output for about 100ns, a relatively long time, when the interpolation clock is switched. Since the unsettled waveforms are directly translated to jitter, during the settling time the output jitter will be increased temporally, after the interpolation clock is switched. This will be called the output jitter increase problem after this.

To solve the output jitter increase problem while maintaining the selective phase interpolation scheme for power and area efficiency, the phase interpolation scheme using four interpolation clocks (PI4C) is proposed. This scheme solves the output jitter increase problem by using two dummy interpolation windows. Fig. 4 depicts the operation of PI4C.

As shown in fig. 4, thirteen clocks generated from phase interpolation are classified into three groups: one selection window and two reserve windows. Clocks generated from two center interpolation clocks among four interpolation clocks consists the selection window. Similarly, clocks generated from two left interpolation clocks and two right interpolation clocks consist the left and right reserve windows, respectively. The selection window is the only window where the output clock is selected. In contrast, the reserve window is a kind of dummy window where nothing is selected. Fig. 5 explains how this scheme solves the output jitter increase problem when the interpolation clock is switched.

When the successive UP (or DOWN) signal requires the output clock to be selected from outside the selection window, the selection window and two reserve windows should be shifted left (or right). To shift the selection window and two reserve windows left, left reserve window and the selection window are changed into the selection window and right reserve window respectively. At the same time one of the four interpolation clocks, which was used in generation of previous right reserve window, is changed, creating new left reserve window.

During this process, phase re-interpolation only occurs in one window, the left (or right) reserve window, and in the selection window, the re-interpolation is never happened. Thus, even though the settling of phase interpolation in newly created reserve window takes relatively long settling time, the output clock, which is selected from the selection window, does not suffer from the jitter increase problem. Since this scheme uses only four interpolation clocks, it shows good power and area efficiency. At the same time, it solved the output jitter increase problem, which occurs in PI2C.

In implementing the PI4C, selecting four adjacent interpolation clocks and defining the reserve and the selection window are not simple. If all the four interpolation clocks were shifted left ($\phi 0$, $\phi 1$, $\phi 2$, $\phi 3$ → $\phi 31$, $\phi 0$, $\phi 1$, $\phi 2$) for the change of the selection window, the generation of two reserve windows would be in vain because all windows, including the selection window, should be re-generated and re-interpolate phases. Thus, to avoid this situation, only one interpolation clock, which is concerned with generation of one reserve window, should be changed, while the other clocks

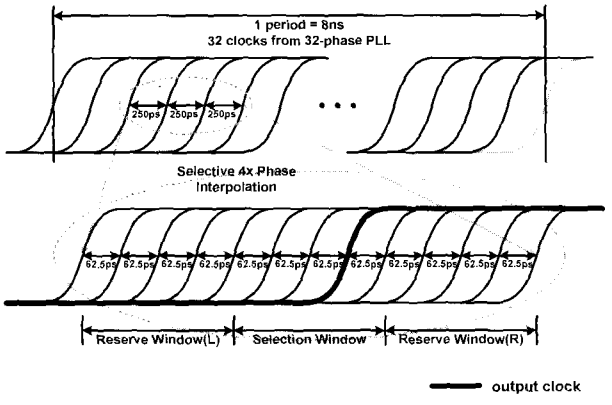


Fig. 4. Phase interpolation scheme using four interpolation clocks (PI4C)

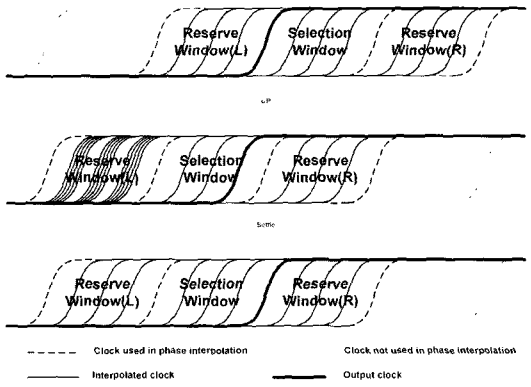


Fig. 5. Selection window shift in PI4C

should remain unchanged to maintain its interpolation results. Fig.6 explains how this operation has been realized.

As shown in fig. 6, four 8-to-1 MUX's choose four interpolation clocks among the 32 clocks from 16-stage differential VCO. A 4x phase interpolator interpolates phases between the four interpolation clocks, totally generating 16 clocks. Among the 16 clocks only 13 clocks that are generated from adjacent interpolation clocks are valid and used in formation of two reserve windows and one selection window. The other 3 clocks generated form non-adjacent interpolation clocks are not used. (Since they are invalid, they are not depicted in fig. 6. The blank space between $\phi 3$ and $\phi 0$, $\phi 2$ and $\phi 31$, and $\phi 1$ and $\phi 30$ are the place where those clocks are exist.) Among the generated 16 clocks, one located within the selection window is selected as an output clock through 16-to-1 MUX.

As the UP/DOWN signal arises, the 16-to-1 MUX changes its selection within the selection window. When the situation that the selection window must be shifted due to the successive UP/DOWN signal incoming arises, one interpolation clock used in the generation of a reserve window is changed. (In fig. 6, the change of $\phi 3$ → $\phi 31$ and $\phi 2$ → $\phi 30$ shows this situation.) In this time, since all MUX's except one maintain their output, one reserve window and the selection window formed by using 3 unchanged interpolation clocks will hold their values. The change of one interpolation clock will make two windows change their values, requiring relatively long settling time, 100ns. However, since those two windows are invalid window and a reserve window where no clock is selected as an output clock, the unsettled results will not be appeared in the output.

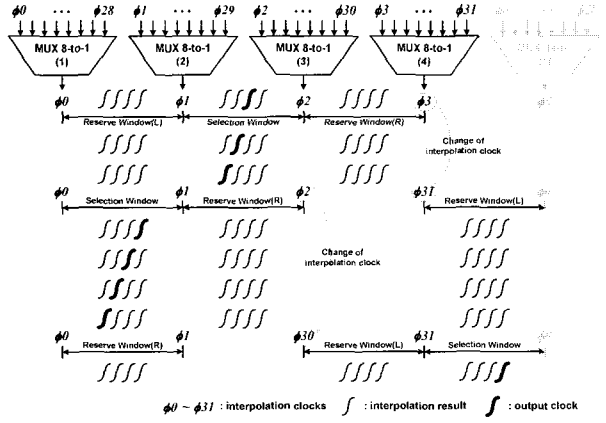


Fig. 6. Detailed PI4C operation according to successive UP signal incomings

III. Clock Feedback Structure

The PLL uses MUX's to select four interpolation clocks and one output clock to interpolate phases and choose an output clock with proper phase. For generation of MUX selection signals, a finite state machine (FSM) synchronized with a clock was employed in the PLL.

When an UP/DOWN signal is occurred, the FSM changes the values of MUX selection signals to shift the output phase. Since the MUX operates asynchronously, the MUX changes its selection as soon as the MUX selection signals are changed. Thus, the output clock phase shifting point corresponds to the MUX selection signal changing point. Since the MUX selection signal is generated from FSM and FSM is synchronized with FSM clock, consequently, the output phase shifting point is decided by the FSM clock's phase.

The output phase shifting point should be carefully determined considering the output clock's state. Since the output clock has discrete phase intervals, output phase shifting at the output clock's rising or falling state can produce an output glitch. Fig. 7 shows how the glitch occurs.

When the FSM clock has a fixed phase, such glitches are unavoidable. Because the FSM clock and the output clock are independent, one cannot predict in what state of output clock the phase shifting will take place and control the phase shifting point according to the output

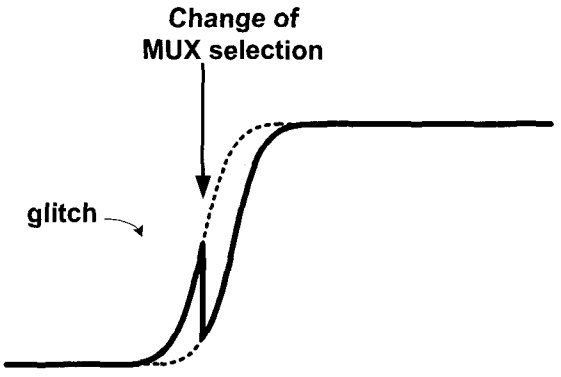


Fig. 7. The glitch occurrence in the output clock due to the phase shifting

clock's state.

In contrast, when the FSM clock phase and the output clock maintain a constant phase difference, one can easily predict and control the output phase shifting point. Maintaining constant phase difference between output clock and FSM clock is possible by feeding the output clock to the FSM clock with delay. Thus, proposed PLL feed the output clock back to the FSM clock and by tuning the delay force the output phase shifting point to place out of the output clock's rising or falling state. By adopting this approach, the PLL can eliminate the possible output glitches, which can harm the system stability.

IV. The Architecture Of The PLL

The implemented PLL is composed of one 32-phase PLL, four 8-to-1 MUX's, a 4x phase interpolator, a 16-to-1 MUX, and a FSM. Fig. 8 shows the architecture of the PLL.

The 32-phase PLL generates 32 clocks with regular phase intervals, by using the 16-stage differential VCO. Among the 32 clocks, four clocks are selected through four 8-to-1 MUX's as interpolation clocks. The four interpolation clocks are fed to 4x phase interpolator and the interpolator multiplies the number of phases by four, improving the phase resolution to 1/128 of the clock period. Then, one clock with required phase is selected among the 16 clocks by a 16-to-1 MUX. The FSM manages all the processes by generating MUX selection signals, with regard to the incoming UP/DOWN signals.

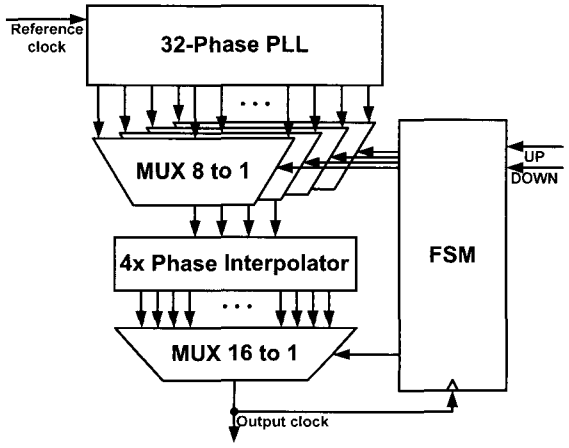


Fig. 8. The architecture of the PLL

Note that the output clock is fed back to the FSM to assure there is no output glitch occurrence due to the output phase shifting.

To multiply the phase resolution by the factor of four, an interpolator cell that doubles the phase resolution is cascaded twice. The phase interpolation is achieved by wiring the output of two adjacent clocks [2]. Several simulation results report that the interpolator cell [2] should be asymmetrically weighted (40% for leading phase and 60% for lagging phase), to place the interpolated clock phase in the middle of two adjacent clock phases. This asymmetric structure is sensitive to device mismatch and this increases the risk of a misalignment of the interpolated phase. Thus, an interpolator cell, which has a symmetric structure [3], is adopted in the PLL to eliminate the risk.

V. Measurements

Fig. 9 shows the microphotograph of the PLL. Since the PLL was designed for 4-channel communication, all parts except the 32-phase PLL were repeated four times. The chip size of 4-channel PLL was 0.84mm^2.

To find out whether the jitter due to the interpolation clock switching is eliminated or not, five successive UP clock was assigned externally. Since the interpolation clock switching occurs every four successive UP/DOWN signal incomings, the results can directly show whether the jitter is increased due to the interpolation clock switching or not. Fig. 10 shows the shifted output phases due to the five successive UP signal incomings.

As shown in fig. 10, despite of the switching of the

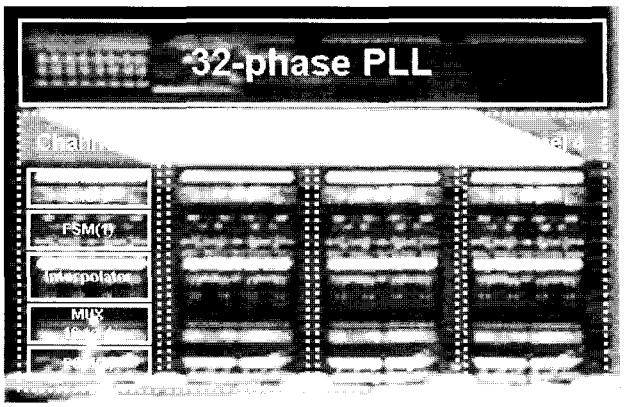


Fig. 9. Microphotograph of the implemented PLL

interpolation clock, no additional jitter was introduced in the output. Thus, it can be inferred from the result that the PI4C eliminated the output jitter increase problem.

Fig. 11 shows the PLL jitter histogram. The PLL presents very low jitter performance –the peak-to-peak jitter was 31ps, which corresponds to 0.4% of the output clock period.

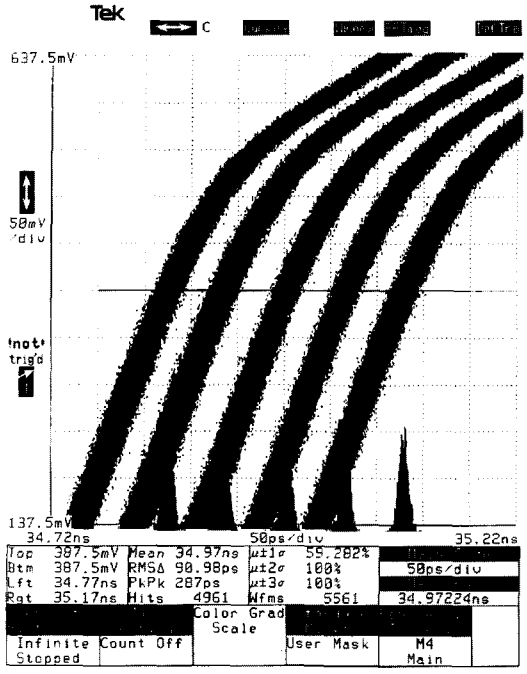


Fig. 10. Output clock phase shifting according to five continuous UP signal occurrences.

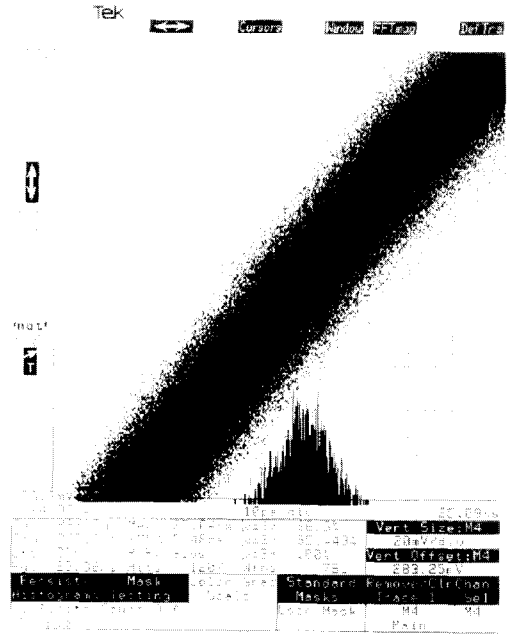


Fig. 11. PLL jitter histogram

Table 1. Implemented PLL performance

Technology		0.25μm, CMOS
Reference clock speed		25MHz
Operation speed		125MHz
Phase resolution		62.5ps (128 phases)
32-phase PLL	VCO gain	160MHz/V
	PM	74°
	BW	560kHz
	Lock Range	10~39MHz(Ref. clock)
Chip area		0.84mm^2 (4-channel)
Power consumption		80mW (4-channel)

Table 1 summarizes the performance of the PLL. It was fabricated in a 0.25μm CMOS technology with 2.5V supply voltage.

VI. Conclusion

An 125MHz PLL having 128-phase resolution was designed for digital timing recovery. To achieve the 128-phase resolution, a phase interpolation technique was applied. A scheme that interpolates phases over only selected windows instead of overall period was adopted to improve power and area efficiency. To avoid output jitter increase problem, which appears when the interpolation clock is switched, four interpolation clocks were used, creating one selection window and two reserve windows. Due to the two reserve windows that perform phase interpolation in advance saving time for settling of interpolation no additional jitter was introduced in output when the interpolation clock is switched.

Also, the PLL adopted clock feedback structure, which feeds the output clock back to the FSM, to assure that no glitch appears in output due to the output phase shifting. Without this, if the fixed phase clock is fed to FSM, an output glitch can occur when the MUX selection is changed during the output clock's rising or falling. Through the clock feedback structure, one can force the MUX selection change to occur when the output clock is in its flat state – not in its rising or falling state – by tuning the delay to FSM and can eliminate the possibility of output glitch occurrence.

The PLL was fabricated in 0.25μm CMOS technology and occupies area of 0.84mm^2 and consumes power of 80mW under 2.5V supply. The experimental result

proved that the PLL do not introduce any additional jitter when the interpolation clock is switched. The PLL presented very good jitter performance – the peak-to-peak jitter was 31ps, which is only 0.4% of the clock period.

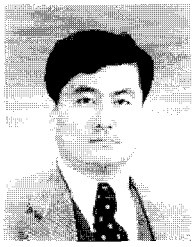
REFERENCES

[1] S. Sidiropoluos *et al.*, A semi-digital DLL, IEEE J. Solid-State Circuits, vol. 32, pp. 1683-1692, Nov. 1999

[2] B.W. Garlepp *et al.*, "A Portable DLL architecture for CMOS Interface Circuits", *Symp. On VLSI Circuits*, pp.214-215. Jun. 1998

[3] K. Nakamura *et al.*, A CMOS 50% duty cycle repeater using complementary phase blending, IEEE Symp. VLSI Circuits Dig. Tech. Papers, pp. 48-49, June. 2000

Hayun Chung was born in Iowa City, Iowa in 1979. She received the B.S. degree from the School of Electrical Engineering, Seoul National University, Seoul, Korea, in 2002, where she is currently working toward the M.S. degree. She has worked on architectures and CMOS circuits for high-speed I/O interfaces. Her current research interests include high-speed CMOS circuits and communication IC's.



Deog-Kyoon Jeong received the B.S. and M.S. degrees in electronics engineering from Seoul National University, Seoul, Korea, in 1981 and 1984, respectively, and the Ph.D. degree in electrical engineering and computer sciences from the University of California, Berkeley, in 1989.

From 1989 to 1991, he was with Texas Instruments Inc., Dallas, TX, where he was a Member of Technical Staff and worked on the modeling and design of BiCMOS gates and the single-chip implementation of the SPARC architecture. He joined the faculty of the Department of Electronics Engineering and Inter-University Semiconductor Research Center, Seoul National University, as an Assistant Professor in 1991. He is currently a Professor of the School of Electrical Engineering, Seoul National University. His main research interests include high-speed I/O circuits, VLSI systems design, microprocessor architectures, and memory systems.

Wonchan Kim was born in Seoul, Korea, on December 11, 1945. He received the B.S. degree in electronics engineering from Seoul National University, Korea, in 1972. He received the Dip.-Ing. and Dr.-Ing. degrees in electrical engineering from the Technische Hochschule Aachen, Aachen, Germany, in 1976 and 1981, respectively. In 1972, he was with Fairchild Semiconductor Korea as a Process Engineer. From 1976 to 1982, he was with the Institut fur Theoretische Electrotecnik RWTH Aachen. Since 1982, he has been with the School of Electrical Engineering, Seoul National University, where he is currently a Professor. His research interests include development of semiconductor device and design of analog/digital circuits.