A 5-Gb/s ADC-Based Feed-Forward CDR in 65 nm CMOS

Oleksiy Tyshchenko, Ali Sheikholeslami, *Senior Member, IEEE*, Hirotaka Tamura, *Member, IEEE*, Masaya Kibune, Hisakatsu Yamaguchi, and Junji Ogawa, *Member, IEEE*

***Abstract*—This paper presents an ADC-based CDR that blindly samples the received signal at twice the data rate and uses these samples to directly estimate the locations of zero crossings for the purpose of clock and data recovery. We successfully confirmed the operation of the proposed CDR architecture at 5 Gb/s. The receiver is implemented in 65 nm CMOS, occupies 0.51 mm^2, and consumes 178.4 mW at 5 Gb/s.**

***Index Terms*—Clock and data recovery, CDR, ADC-based CDR, feed-forward CDR, blind-sampling CDR, all-digital CDR.**

I. Introduction

COMMUNICATION standards reflect the growing demand for higher data rates in wireline channels. The rapidly evolving integrated circuit (IC) technologies enable the transceivers to keep up with these high data rates. The advancement of the channels, however, typically lags the advancement of the IC technologies. As a result, current multi-Gb/s standards require the transceivers to operate in the presence of high signal attenuation.

Receivers with binary samplers typically compensate the received signal for channel loss in the analog domain, prior to sampling, using feed-forward equalization (FFE), decision feedback equalization (DFE) or both. This analog equalization limits the amount of channel compensation that can practically be implemented in an integrated receiver. Replacing the binary sampler with an analog-to-digital converter (ADC), as shown in Fig. 1, allows integration of extensive digital signal processing (DSP) into the receiver to compensate for high channel distortion after the signal is sampled.

Recently reported ADC-based clock and data recovery (CDR) circuits align the sampling clock with the received signal using a phase-tracking feedback loop [1]–[4], as shown in Fig. 1(a). This architecture requires a voltage-controlled oscillator (VCO) or a phase interpolator (PI), both analog circuits, to adjust the phase of the sampling clock. To eliminate these analog circuits (and their phase control) in favor of an all-digital implementation, a blind-sampling ADC-based CDR (shown in Fig. 1(b)) samples the received signal without phase locking to the signal. The CDR then interpolates between the blind samples to obtain a new set of samples in order to recover phase and data [5], [6]. The interpolator, however, is relatively complex and since it remains in the CDR loop, it contributes to the loop latency.

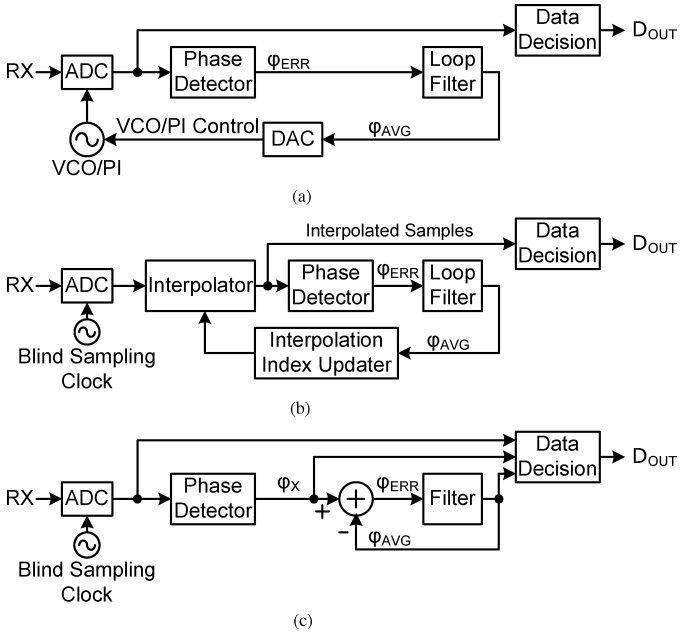


Fig. 1. ADC-based CDR architectures: (a) phase-tracking CDR; (b) blind-sampling interpolating feedback CDR; (c) blind-sampling feed-forward CDR (this work).

In this paper, we propose a feed-forward CDR architecture, shown in Fig. 1(c), that estimates the data phase directly from the blind digital samples, hence eliminating the need for digital interpolation. In this architecture, we use a low-complexity digital phase detector (PD) and data decision circuits along with a digital FFE. We have implemented and characterized the proposed CDR in 65 nm CMOS at 5 Gb/s.

The remainder of this paper is organized as follows. Section II reviews basic concepts of binary and ADC-based sampling in CDRs. Section III presents the proposed feed-forward CDR architecture. Section IV describes the implementation of the digital CDR. Section V validates the proposed architecture through the CDR simulations and the test-chip measurements. Finally, Section VI concludes this paper.

II. Background

A conventional phase-tracking CDR with a binary front-end samples the received signal with a flip-flop and represents every

Manuscript received August 28, 2009; revised January 04, 2010; accepted February 20, 2010. Current version published June 09, 2010. This paper was approved by Associate Editor Jafar Savoj.

O. Tyshchenko and A. Sheikholeslami are with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, Ontario M5S 3G4, Canada (e-mail: tyshche@eecg.utoronto.ca; ali@eecg.utoronto.ca).

H. Tamura, M. Kibune, H. Yamaguchi, and J. Ogawa are with Fujitsu Laboratories, Kawasaki 211-8588, Japan.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSSC.2010.2047156

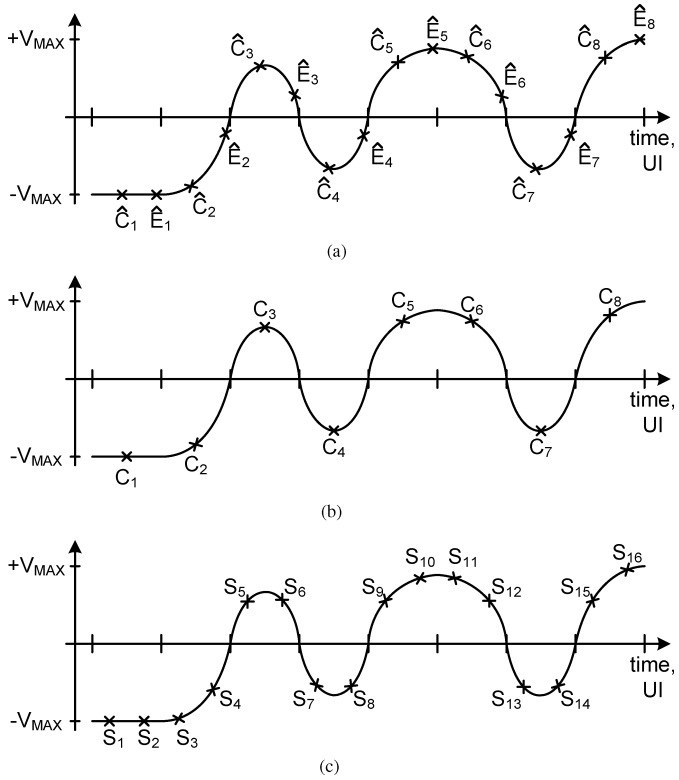


Fig. 2. Binary and ADC-based sampling schemes: (a) phase-tracking binary sampling at $2f_B$; (b) phase-tracking ADC sampling at f_B; (c) blind ADC sampling at $2f_B$.

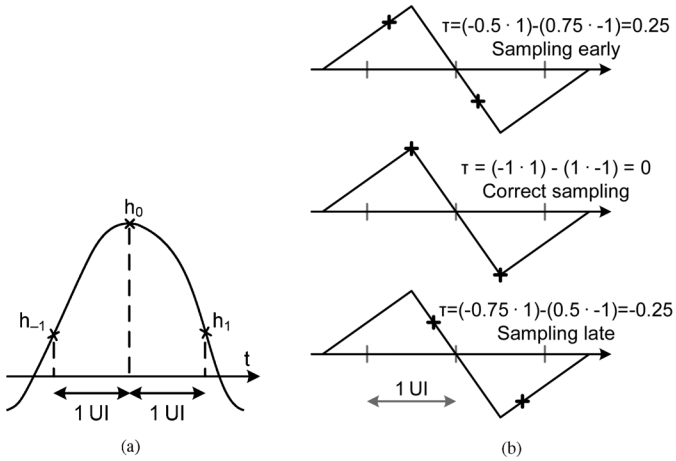


Fig. 3. Mueller-Müller timing recovery scheme. (a) Impulse response. (b) Continuous data.

sample with a single bit. These binary samples preserve the sign of the signal at the sampling instances, but discard the magnitude of the signal. In Fig. 2(a), we use ˆ (*hat*) to denote binary samples. To recover clock and data, the CDR samples the signal at twice the baud rate, $2f_B$: it takes two samples in every unit interval (UI)—one close to UI center, $\hat{C}_i$, and one close to UI edge, $\hat{E}_i$. The samples capture only 2 bits of information for every UI. The CDR then uses the UI edge samples, $\hat{E}_i$, along with their surrounding UI center samples, $\hat{C}_i$ and $\hat{C}_{i+1}$, to drive the phase-tracking feedback loop in order to recover the clock and to align the sampling instances with the received signal. The UI center samples, $\hat{C}_i$, become the recovered data.

An ADC-based CDR, in contrast, samples the signal with an ADC (instead of a flip-flop) and represents every sample with multiple bits, say 5 bits. These digital samples preserve both the sign and the magnitude of the signal, as shown in Figs. 2(b) and (c). The digital samples capture more information about the signal at the sampling instances, compared to binary samples. This extra information in the samples allows the ADC-based CDRs to recover clock and data using either phase-tracking sampling at f_B or blind sampling at $2f_B$, as we will discuss next.

Recently published phase-tracking ADC-based CDRs sample the signal at f_B: they take one sample per UI close to UI center, as shown in Fig. 2(b) [1]–[4]. The CDR then extracts the timing information from the baud-rate samples, C_i, to drive the phase-tracking loop. Mueller-Müller scheme [7] is typically used for this timing recovery. This scheme relies on the timing estimation from the impulse response of the channel in the presence of some inter-symbol interference (ISI). To illustrate this, we use a sample impulse response shown in Fig. 3(a). In this example, h_0 is the signal cursor, while h_{-1} and h_1 are the first pre- and post-cursors of the signal. If the signal is sampled such that $h_{-1} = h_1$, then h_0 is close to the maximum of the impulse response, which is the desirable sampling phase. Hence, a function $\tau = h_{-1} - h_1$ indicates if the sampling phase is early or late for optimum sampling, and it can guide a timing recovery from the baud-rate samples. Mueller and Müller also showed in [7] that a simple operation estimates this timing function from a continuous data stream in the average sense: $\tau = (y_n \cdot \hat{y}_{n-1}) - (y_{n-1} \cdot \hat{y}_n)$. In this equation, y_n and y_{n-1} are the signal samples, while $\hat{y}_n$ and $\hat{y}_{n-1}$ are the corresponding decisions bits. In Fig. 3(b) we illustrate this estimation of timing from continuous data through a simplified example. Since the timing recovery aligns the samples with UI centers, the signs of the samples are the recovered data. In this CDR scheme, the timing function is specific to the channel response. If the pre- and post-cursors in the channel impulse response are asymmetric then the timing function used in our example does not converge. Thus, the Mueller-Müller scheme is not suitable for applications with a wide variety of potential channel responses.

Blind-sampling ADC-based CDRs sample the received signal without aligning the sampling instances to the signal. Out of two common sampling rates, f_B and $2f_B$, blind nature of sampling rules out sampling at f_B, since in the worst case, the baud-rate samples might fall on UI edges, which makes error-free data recovery practically impossible. Hence, typical blind ADC-based CDRs sample the signal at $2f_B$, as illustrated in Fig. 2(c). To recover clock and data, the CDR interpolates between the blind samples, S_i, a new set of two samples per UI—the phase-locked set [5], [6]. In the phase-locked set, one sample is close to UI center and the other sample is close to UI edge. The CDR then uses the interpolated UI edge samples to drive the digital phase recovery loop in order to align the interpolated samples with the received signal. The signs of the interpolated UI center samples become the recovered data.

Since digital samples capture both the sign and the magnitude of the signal, ADC-based CDRs implement the channel equalization after sampling in the digital domain. This post-sampling

digital equalization allows ADC-based CDRs to integrate a higher degree of channel equalization, compared to pre-sampling analog equalization in binary CDRs. This, in turn, makes the ADC-based CDRs preferable over binary CDRs for applications that require a high degree of compensation for channel distortion.

Baud-rate phase-tracking ADC-based CDRs, similar to binary phase-tracking CDRs, require a feedback loop that crosses the analog/digital domain boundaries and contains a phase adjustable clock generator—a VCO or a PI. Blind-sampling ADC-based CDRs implement the phase-tracking loop entirely in the digital domain, preventing the loop from crossing the analog/digital domain boundaries. This all-digital loop implementation comes at the cost of doubling the ADC conversion rate from f_B to $2f_B$, increasing the ADC area and power consumption. The increased sampling rate, on the other hand, allows to eliminate the VCO/PI from the CDR, which reduces the circuit complexity and loop latency, thus increasing the jitter-tracking bandwidth. Furthermore, containing the phase-tracking loop in the digital domain simplifies the design and verification process for the blind-sampling CDRs, and it allows to take full advantage of the IC technology scaling.

The blind-sampling CDR architecture imitates the phase-tracking clock and data recovery by means of digital interpolation. The interpolator in the feedback loop, however, is a relatively complex block, and its latency contributes to the total latency of the digital phase-tracking loop, which has a negative impact on the loop stability.

In the next section, we present our proposed feed-forward ADC-based CDR architecture that eliminates the phase-tracking loop from the CDR. In this architecture, we recover the clock and data directly from the blind digital samples without the need for interpolation, thus reducing the CDR complexity.

III. Proposed CDR Architecture

Fig. 4 presents the block diagram of the proposed blind-sampling ADC-based receiver with feed-forward CDR architecture. We sample the received signal, RX, with two time-interleaved 5-GS/s 5-bit ADCs. A two-phase blind sampling clock triggers the ADCs to take 2 samples per UI. We refer to the phases of the sampling clock as 0° phase and 180° phase. A 2:32 demux then feeds 32 samples with 5-bit resolution at every 16 UI interval to the digital CDR. A 1:16 clock divider divides the sampling clock to trigger the digital CDR. A two-tap FFE compensates the received signal for the channel loss such that both samples in each UI are equalized. A phase detector (PD) uses the equalized samples to estimate the instantaneous zero-crossing phase, ϕ_X, for every UI with a data transition. A phase subtracter with a low-pass filter (LPF) in a feedback loop form the phase-recovery filter of the CDR. This filter uses ϕ_X to generate the average zero-crossing phase, ϕ_{AVG}. Since we use only the input and the output of the phase-recovery filter (and no internal signals), the CDR is of a feed-forward type.

A slicer and a data decision block compose the data decision path of the CDR. In this path, the slicer detects the signs of the equalized samples to represent each sample with a single bit. Then, the data decision block picks one sliced sample per UI as a data bit by comparing the instantaneous phase, ϕ_X, with the average phase, ϕ_{AVG}, for every UI.

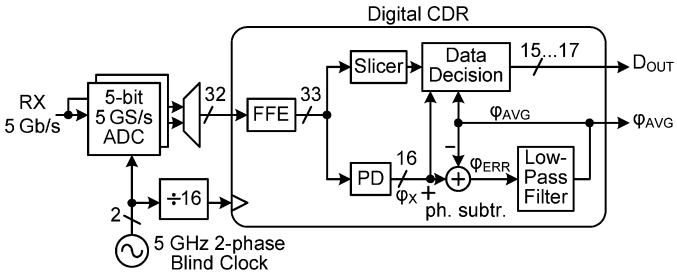


Fig. 4. Block diagram of receiver with feed-forward CDR architecture.

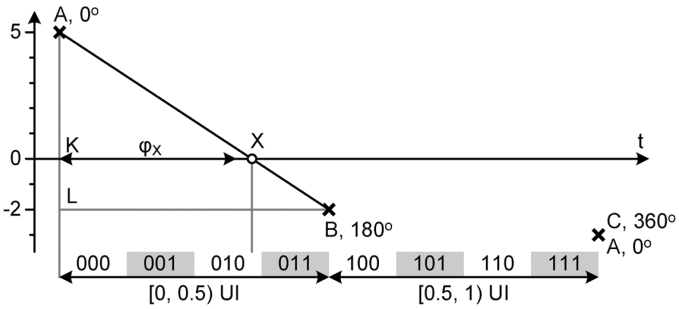


Fig. 5. Linear estimation of instantaneous phase, ϕ_X.

In the following section, we describe the implementation of the digital CDR.

IV. CDR Implementation

A. Phase Detector

The PD estimates the data zero-crossing phase from the equalized digital samples of the received signal. We refer to this phase as the instantaneous zero-crossing phase, ϕ_X. The PD processes the samples from 16 cycles of the sampling clock in parallel (2 samples per cycle) and outputs ϕ_X for the UIs with data transitions.

Fig. 5 illustrates the phase detection scheme through an example of a single cycle of the sampling clock. The PD looks at 3 consecutive samples: A, B and C, which correspond to 0°, 180° and 360° phases of the blind sampling clock. Since sample C corresponds to 360° phase, it is also sample A (0° phase) in the following cycle of the sampling clock.

When two adjacent samples have opposite signs (A and B in our example), the PD linearly estimates the time of zero-crossing between these two samples with respect to 0° phase of the sampling clock. We mark the estimated zero crossing as point X in Fig. 5. We find ϕ_X from two proportional triangles: ABL and AXK. The adjacent samples are 0.5 UI apart in time, which makes side BL of ABL a constant equal to 0.5 UI. Side AL of ABL is the sum of amplitudes of A and B, while side AK of AXK is the amplitude of A. Thus, we find ϕ_X, which is side XK of AXK, as a ratio:

$$\phi_X = \frac{0.5A}{(A-B)}. \tag{1}$$

To maintain low circuit complexity, we limit the accuracy of this division to 2 bits. The flowchart in Fig. 6 shows that this 2-bit

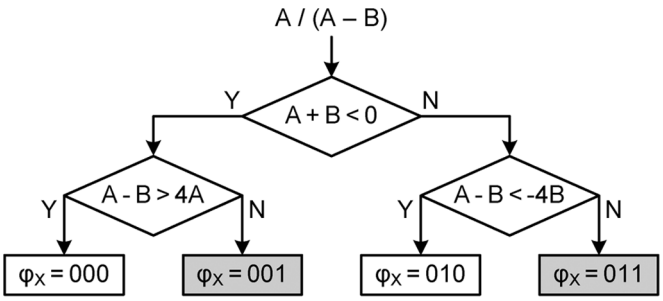


Fig. 6. Flowchart of 2-bit accurate division for calculating ϕ_X.

accurate division requires only simple operations: addition/subtraction and left shift by 2 (multiply by 4 in decimal).

Since A and B are 0.5 UI apart, the total resolution of ϕ_X is 3 bits per UI. When a transition takes place between B and C, (1) becomes

$$\phi_X = 0.5 + \frac{0.5B}{(B-C)} \tag{2}$$

which simply changes the most significant bit (MSB) of ϕ_X from '0' to '1'. We postpone the discussion of the effect of limiting the ϕ_X accuracy to 3 bits till the Data Decision subsection.

Nominally, there is at most one data transition in every cycle of the sampling clock: either between A and B or between B and C. However, duty-cycle distortion (DCD) and frequency offset between transmitter and receiver might cause two transitions per sampling cycle: between A and B as well as between B and C. When two such transitions occur, the PD calculates ϕ_X as modulo-1 sum of both zero-crossing phases so that both transitions contribute to the average phase recovery. This summation allows the phase detection scheme to estimate the data phase in the presence of DCD and frequency offset between transmitter and receiver.

In the following subsection we describe the phase recovery filter that we implemented in the CDR.

B. Phase Recovery Filter

The phase recovery filter averages the instantaneous phase, ϕ_X, to recover the average zero-crossing phase, $\phi_{\rm AVG}$. Similar to PI control in conventional phase-tracking CDRs, $\phi_{\rm AVG}$ in our feed-forward architecture tracks the data phase in an average sense. For this phase tracking, we use a discrete-time IIR filter shown in Fig. 7. The filter consists of a phase subtracter and a 3rd order low-pass filter (LPF) in a feedback loop.

The phase subtracter, shown in the left inset of Fig. 7, calculates the phase difference between ϕ_X and $\phi_{\rm AVG}$ for 16 UIs at a time and outputs the combined phase error, $\phi_{\rm ERR}$, for these 16 UIs. To assure that the phase recovery converges for any offset between ϕ_X and $\phi_{\rm AVG}$, we calculate $\phi_{\rm ERR}$ in a modulo manner such that ϕ_{ERR_i} is in the range $[-0.5, 0.5)$ UI. The subtracter excludes the UIs without data transitions from contributing to $\phi_{\rm ERR}$. $\phi_{\rm ERR}$ in our architecture plays the same role as the PD output in a conventional phase-tracking CDR.

We feed $\phi_{\rm ERR}$ into the 3rd order LPF, which consists of three cascaded discrete-time delaying integrators with gains K_1, K_2 and K_3. There are three forward paths in the LPF: 1st, 2nd and 3rd order paths (see the right inset in Fig. 7). These three paths

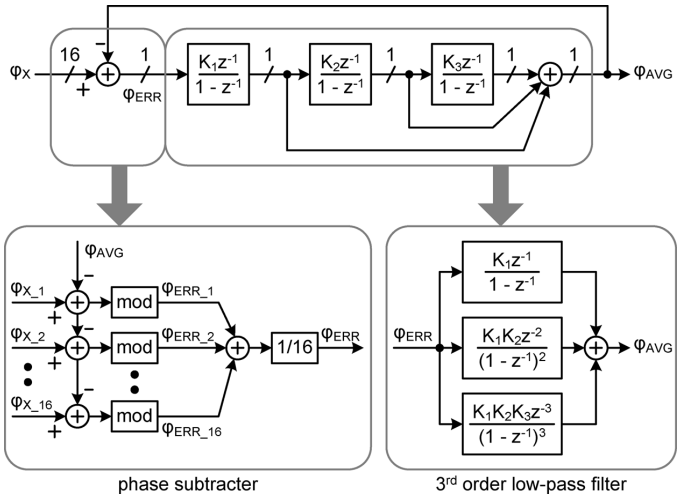


Fig. 7. Phase recovery filter.

add up to the average (recovered) phase, $\phi_{\rm AVG}$. The transfer function of the entire phase recovery filter is

$$\frac{\phi_{AVG}}{\phi_X} = \frac{A_{FW}}{1 + A_{FW}} \tag{3}$$

where A_{FW} is the forward gain of the LPF:

$$A_{FW} = \frac{K_1 z^{-1}}{1 - z^{-1}} + \frac{K_1 K_2 z^{-2}}{(1 - z^{-1})^2} + \frac{K_1 K_2 K_3 z^{-3}}{(1 - z^{-1})^3}. \tag{4}$$

Three criteria determine the filter gain values: the desired jitter-tracking bandwidth of the CDR, the absence of gain peaking in the jitter-transfer function of (3), and the low-circuit-complexity filter implementation. First, we choose the CDR jitter-tracking bandwidth (approximately 5 MHz in the proposed receiver). Then, we determine through simulations the gain values that achieve this bandwidth while minimizing the gain peaking in the jitter transfer function. Finally, we round-off the gain values to the nearest easy-to-implement values in binary. This procedure leads to $K_1 = 3/64$, $K_2 = 7/2048$, and $K_3 = 5/2048$. To illustrate the low complexity gain implementation, $K_1 = 3/64$ is implemented as $K_1 = 1/32 + 1/64$, where gains of $1/32$ and $1/64$ are obtained through right-shifting the input value by 5 and 6 bits. In a similar manner, K_2 and K_3 are composed of right-shift and addition operations to maintain the low circuit complexity. We used these gain values in the simulations and measurements presented in Section V.

To explore the effect of the order of the phase-recovery filter on the CDR performance, we reduced the filter order from 3rd to 2nd and 1st, and simulated the CDR's jitter tolerance, as illustrated in Fig. 8. Note that in all three cases, the CDR's jitter-tracking bandwidth remains constant. As the order changes from 1st to 2nd, the high-frequency jitter tolerance improves by approximately 0.2 $\rm UI_{PP}$. Furthermore, with the use of the 2nd order filter, the jitter tolerance roll-off slope increases allowing for a higher tolerance at low frequencies. The 3rd order filter shows a small improvement of the jitter tolerance compared to the 2nd order filter: the high-frequency jitter tolerance remains unchanged, but the low-frequency jitter tolerance increases by up to 3× (at 32 kHz for instance). For a safe design with a high

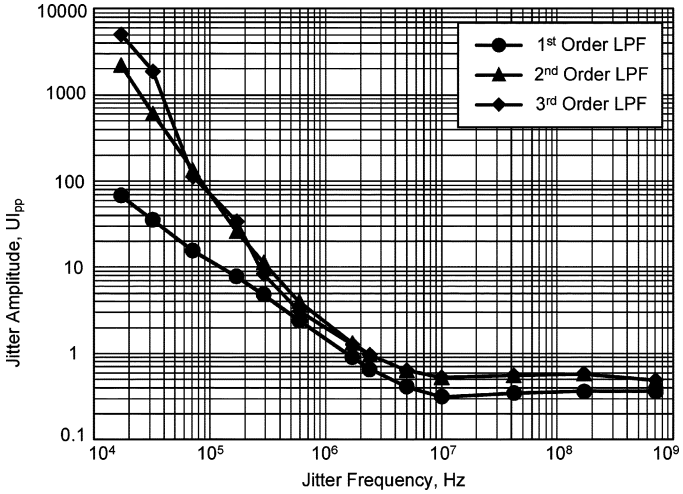


Fig. 8. Jitter tolerance dependence on the LPF order (simulated).

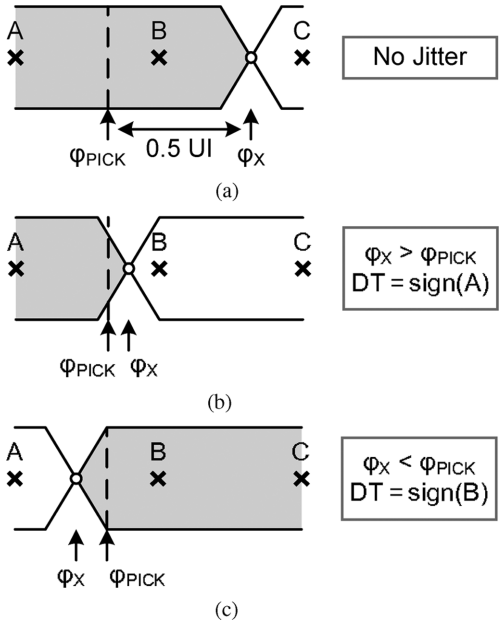


Fig. 9. Data decision scheme. (a) Jitter-free case; (b) jitter example 1; (c) jitter example 2.

tolerance to low-frequency jitter, we use the 3rd order filter in the proposed feed-forward CDR.

The resolution of the intermediate phase values in the integrators is 16 bits: 10 least significant bits represent the fractional part of the phase (1 UI long period), while 6 most significant bits represent the integer part. To tolerate the jitter exceeding 64 UIs (2^6 UIs), we use 'roll-over' rather than 'saturating' counters in the integrators.

The CDR uses the recovered $\phi_{\rm AVG}$ along with ϕ_X for data decision according to the scheme that we present in the following subsection.

C. Data Decision

The proposed feed-forward clock recovery eliminates the interpolator from the CDR thus reducing the circuit complexity. As a consequence of this interpolator elimination, the value of the signal at the UI center is not interpolated and therefore is unknown. To enable error-free data recovery along with the feed-forward clock recovery, a data decision scheme is essential to the proposed feed-forward CDR. The role of the data decision block is to estimate the sign of the received signal near the maximum eye opening, i.e., near the UI center. Since $\phi_{\rm AVG}$ indicates the average position of the UI boundaries, we calculate the position of the UI centers by adding 0.5 UI to $\phi_{\rm AVG}$ using modulo-1 addition. We refer to this UI center phase as the data-picking phase, $\phi_{\rm PICK}$. The data decision block takes the sliced samples from 16 sampling cycles and picks one decision sample for every UI by comparing ϕ_X and $\phi_{\rm PICK}$.

Fig. 9 illustrates the data-picking scheme through an example of a single sampling cycle. The data decision block takes three consecutive sliced samples (A, B and C) and picks one of these samples as the decision bit. This decision bit is picked such that it is close to the UI center. We will now explain how this scheme recovers data in a jitter-free case and in the presence of jitter.

In a jitter-free case shown in Fig. 9(a), ϕ_X coincides with $\phi_{\rm AVG}$, and hence it is 0.5 UI away from $\phi_{\rm PICK}$. $\phi_{\rm PICK}$ in fact points to the UI in which we are recovering the data (shaded in the figure). The decision scheme thus picks one of the two samples adjacent to $\phi_{\rm PICK}$: either A or B in our example. In a jitter-free case, both samples adjacent to $\phi_{\rm PICK}$ have the same sign and hence the decision is trivial.

In the presence of jitter, the two samples adjacent to $\phi_{\rm PICK}$ might belong to different UIs, and these samples might have opposite signs, as we illustrate in Figs. 9(b) and (c). In this case, the data decision scheme picks the sample that belongs to the same UI to which $\phi_{\rm PICK}$ points (shaded UI in the figure). For instance, in Fig. 9(b) the jitter causes ϕ_X to shift left compared to Fig. 9(a) and we pick A as the decision data. In the example of Fig. 9(c), ϕ_X shifts right compared to Fig. 9(a) and we pick sample B. This scheme requires a single comparison between ϕ_X and $\phi_{\rm PICK}$ for every UI.

Simulations revealed that jitter, limited channel bandwidth and duty cycle distortion (DCD) reduce the width of UI-long data pulses and thus cause two transitions per sampling cycle. We refer to this case as an *isolated pulse*. Fig. 10 illustrates a nominal and isolated UI-long pulses. In the nominal case of Fig. 10(a), samples B and C are equidistant from $\phi_{\rm PICK}$, which makes both samples equally correct decisions. However, in the presence of isolated pulses (Figs. 10(b) and (c)), two transitions per UI prohibit defining a single instantaneous phase value, ϕ_X. As a consequence, a comparison between ϕ_X and $\phi_{\rm PICK}$ proves insufficient for a correct data decision. The data-picking scheme detects these isolated pulses using XOR operation on every pair of consecutive samples. It then disregards the phase information and picks the sample at the center of the pulse, i.e., farthest from both transitions. When two transitions happen in the same sampling cycle between A and B, and between B and C (see Fig. 10(b)), we pick B as the decision data. In a similar manner we check for isolated pulses at the boundary between two consecutive sampling cycles. As Fig. 10(c) illustrates, when we detect a transition between $\rm B_{\it i}$ and $\rm C_{\it i}$ in sampling cycle i, and between $\rm A_{\it i+1}$ and $\rm B_{\it i+1}$ in cycle $i+1$, we pick $\rm C_{\it i}$ ($\rm A_{\it i+1}$) as the decision bit.

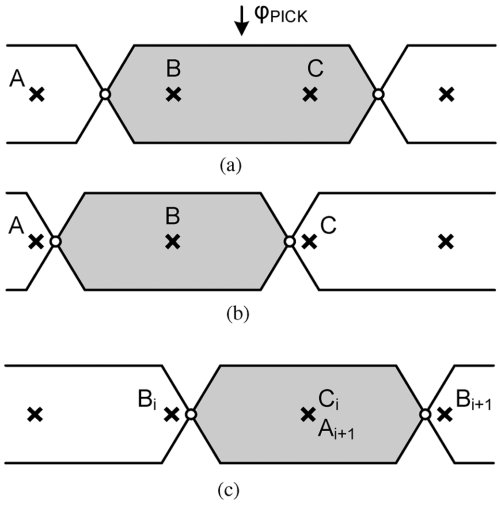


Fig. 10. Data decision with isolated pulses. (a) Nominal pulse width, jitter-free case; (b) two transitions in the same cycle; (c) two transitions in adjacent cycles.

The proposed data decision scheme based on the comparison between ϕ_X and ϕ_{PICK} recovers the data correctly when ϕ_X deviates by up to 0.5 UI in either direction. Hence, the CDR has the theoretical maximum jitter tolerance of 1 $\mathrm{UI_{PP}}$ at high frequencies. Estimating ϕ_X with 3-bit accuracy (instead of infinite accuracy) results in the reduction of the high frequency tolerance by only 1/8 UI. We postpone further discussion of the jitter tolerance till Section V.

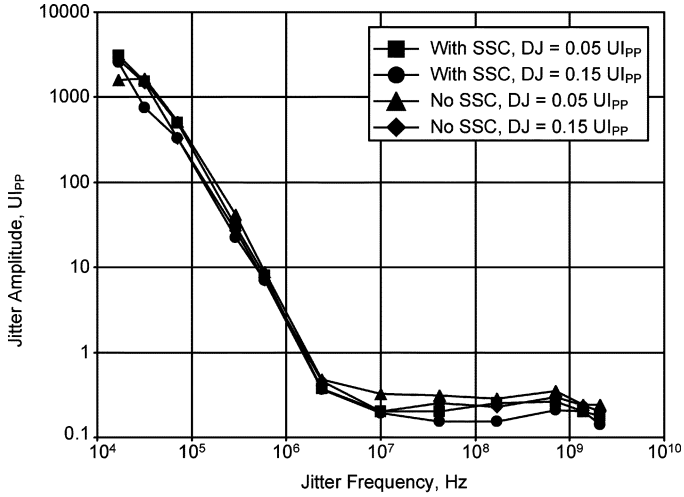
To prevent data errors due to a frequency offset between the transmitter and the receiver, we rely on flow control at the data link layer of the communication protocol. We discuss the frequency offset compensation scheme next.

D. Compensation for Frequency Offset

The transmitter clock determines the data rate at the input of the CDR, while the blind sampling clock determines the data rate at the output of the CDR. Since these two clocks are free-running with respect to each other, a frequency offset between them is inevitable. This frequency offset, in turn, leads to a mismatch between the data rates at the input and at the output of the CDR. A flow control technique compensates for this rate mismatch at the data link layer of the communication protocol.

A small frequency offset between the transmitter and the receiver clocks causes the recovered average phase, ϕ_{AVG}, to constantly shift in one direction. For instance, if the transmitter clock has a higher frequency than the receiver clock, ϕ_{AVG} constantly reduces, indicating that the received UI is shorter than the period of the sampling clock. Thus, ϕ_{AVG}, as well as ϕ_{PICK}, can be used as an indicator of the frequency offset. In our CDR, we use ϕ_{PICK} for this purpose. When ϕ_{PICK} crosses the UI boundaries, the data decision block outputs 15 or 17 valid bits, while it outputs 16 valid bits when ϕ_{PICK} remains within the UI boundaries.

Multiple instances of 17 valid data bits at the CDR output (instead of nominal 16 bits) eventually leads to a data overflow at the data link layer of the protocol. In this case, the flow control reduces the data flow rate from the transmitter by reducing transmitted data window size. This approach allows the physical layer of the protocol (the CDR) to eliminate the generation of the recovered clock and hence to reduce the CDR complexity.



Simulation Conditions	
Input	5 Gb/s, $2^{31}-1$ PRBS
Channel loss	13 dB @ 2.5 GHz
T_{SIM} (f_{JIT} > 250 kHz)	$2x10^5$ UIs
T_{SIM} (f_{JIT} ≤ 250 kHz)	1 jitter period
BER	≤ $5x10^{-6}$
TX pre-emphasis	3 dB
Tx-Rx Δf_{CLK}	600 ppm (nominal)
SSC freq. modulation	0...-5000 ppm @ 32 kHz
Tx-Rx Δf_{CLK} (with SSC)	10600 ppm (max)
Tx RJ	0.17 $\mathrm{UI_{PP}}$ (Gaussian)
Tx DJ	0.19 $\mathrm{UI_{PP}}$ (dual-Dirac)
Rx RJ	0.23 $\mathrm{UI_{PP}}$ (Gaussian)
Rx DJ	see legend (dual-Dirac)

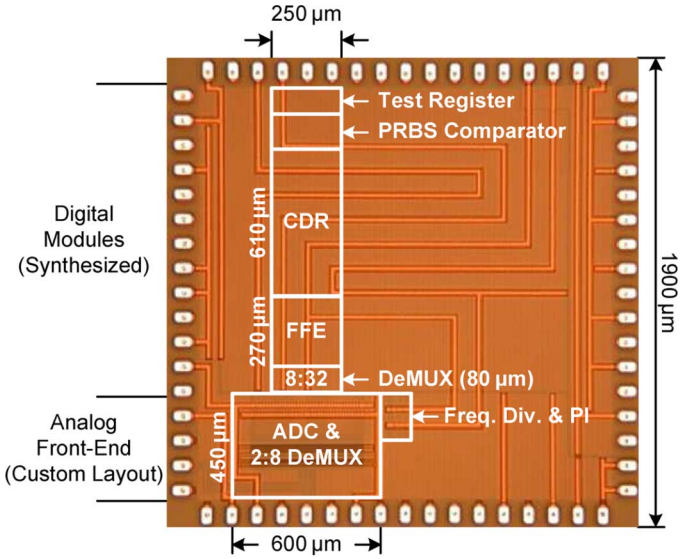
Fig. 11. Simulated jitter tolerance.

The following section presents the results of the CDR simulations and of the test-chip measurements.

V. Simulation and Measurement Results

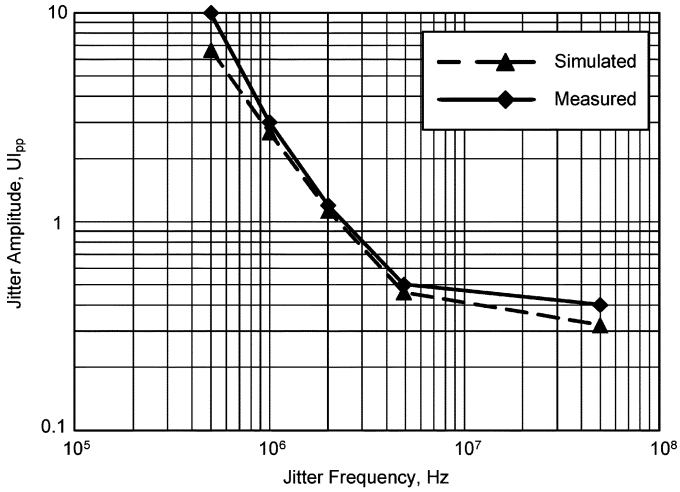
To validate the proposed CDR architecture, we first simulated the CDR on a behavioral level, and then we fabricated and characterized a receiver with this CDR in 65 nm CMOS.

To simulate the CDR, we used an event-driven behavioral model [9] in Simulink. This model accounts for limited channel BW, supports asynchronous clock domains, and allows adding multiple jitter sources into the simulation. Fig. 11 presents the simulated jitter tolerance and summarizes the simulation conditions. In these simulations, we superimposed sinusoidal jitter on random, deterministic jitter (RJ and DJ) and a frequency offset between the transmitter and receiver. We simulated the CDR with a 5 Gb/s $2^{31}-1$ PRBS sequence passed through a channel with 13 dB attenuation at 2.5 GHz. The transmitter pre-emphasis is 3 dB. To maintain reasonable simulation time, we ran the simulations for $2\cdot 10^5$ UIs for jitter frequencies, f_{JIT}, above 250 kHz, and for 1 jitter period for f_{JIT} below 250 kHz, which corresponds to $\mathrm{BER} \leq 5\cdot 10^{-6}$. We set a nominal frequency offset between transmitter and receiver clocks, Δf_{CLK}, to 600 ppm. In addition to this nominal value, we added Δf_{CLK} due to spread-spectrum clocking (SSC) of up to 5000 ppm at 32 kHz. This SSC-induced offset was added both at the transmitter and at the receiver for a total Δf_{CLK} of up to 10600 ppm. The variance



Process	65 nm CMOS
Data Rate	5 Gb/s
Supply	1.2 V
Power	178.4 mW
Receiver Area	0.51 mm^2

Fig. 12. Receiver test-chip photo.



Measurement and Simulation Conditions	
Input	5 Gb/s, 2^7-1 PRBS
Channel loss	10 dB @ 2.5 GHz
T_{SIM} (sim)	5x10^5 UIs
BER (sim)	≤ 2x10^{-6}
BER (meas)	≤ 10^{-12}
TX pre-emphasis	3 dB
Tx-Rx Δf_{CLK}	0
SSC freq. modulation	0
Tx-Rx Δf_{CLK} (with SSC)	0
Tx RJ (sim)	0 UI_{PP}
Tx DJ (sim)	0.05 UI_{PP} (dual-Dirac)
Rx RJ (sim)	0.25 UI_{PP} (Gaussian)
Rx DJ (sim)	0.05 UI_{pp} (dual-Dirac)

Fig. 13. Measured jitter tolerance.

of RJ was adjusted to reach the reported peak-to-peak values within each simulation run. These simulations confirm that the proposed CDR recovers error-free data at 5 Gb/s in the presence of jitter on the transmitter and the receiver sides, frequency offset up to 1.06%, and channel loss of 13 dB at 2.5 GHz.

We implemented a receiver with the proposed CDR architecture in 65 nm standard-logic CMOS. Fig. 12 presents the die photo of the fabricated receiver. The ADC, frequency divider, PI and the 2:8 portion of 2:32 demux are analog custom-designed blocks. The 8:32 portion of 2:32 demux, the FFE, CDR and the test structures (PRBS comparator and test register) are all synthesized.

The ADC consists of two time-interleaved 5 GS/s 5-bit interpolating flash ADCs to achieve the total sampling rate of 10 GS/s. To reduce the receiver input loading, the ADC evaluates four most significant bits (MSBs) using 17 comparators at the front end, and resistively interpolates the least significant bit (LSB) to achieve the 5-bit resolution. The ADC has a measured ENOB of 4.2 bit and a power consumption of 110 mW. After the signal samples are demuxed, we compensate the samples for the channel loss using a half-UI-spaced 2-tap FIR filter as an FFE. The filter tap coefficients are programmable through a serial shift register. The FFE compensates for up to 15 dB of channel attenuation at 2.5 GHz. Since both unequalized and equalized samples are available in the digital domain, the FFE adaptation algorithm can be implemented entirely in the digital domain. The FFE compensates the blind samples for the channel attenuation prior to the CDR, which makes the equalization independent of the clock and data recovery.

Fig. 13 presents the measured jitter tolerance of the fabricated receiver and summarizes the measurement conditions. In these measurements, we used a 2^7-1 PRBS sequence running at 5 Gb/s as the data source. The channel attenuation is 10 dB at 2.5 GHz. We used a 3 dB pre-emphasis at the transmitter with the launch amplitude of 750 mV_{PP}. The measured jitter tolerance was recorded at $\mathrm{BER} \leq 10^{-12}$. For a comparison between the simulated and measured results, we included a simulated jitter tolerance with the same data source and channel loss into Fig. 13. The measured and simulated jitter tolerances closely match each other. The receiver consumes 178.4 mW at 5 Gb/s, including the ADC. The entire receiver occupies the chip area of 0.51 mm^2 (test structures excluded).

VI. Conclusion

We presented a blind-sampling ADC-based feed-forward CDR architecture. In this architecture, we sample the received signal blindly with an ADC at twice the baud-rate. The blind sampling allows removing the phase-tracking feedback loop from the CDR, thus simplifying the receiver architecture. We recover the data phase directly from digital signal samples in a feed-forward manner, hence eliminating the need for a digital interpolating feedback loop. This feed-forward topology reduces the CDR circuit complexity compared to blind-sampling interpolating CDRs.

We fabricated a receiver with this feed-forward architecture in 65 nm CMOS. The test-chip successfully recovers data at 5 Gb/s in the presence of channel attenuation of 10 dB at 2.5 GHz. The receiver occupies 0.51 mm^2 of die area and consumes 178.4 mW of power. Our CDR simulations and test-chip measurements confirm that the proposed architecture is suitable for high-speed serial links.

ACKNOWLEDGMENT

The authors thank Chihiro Sannomiya for her assistance with test-chip design and verification.

REFERENCES

[1] J. Cao *et al.*, "A 500 mW digitally calibrated AFE in 65 nm CMOS for 10 Gb/s serial links over backplane and multimode fiber," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, Feb. 2009, pp. 370–371.

[2] O. Agazzi *et al.*, "A 90 nm CMOS DSP MLSD transceiver with integrated AFE for electronic dispersion compensation of multimode optical fibers at 10 Gb/s," *IEEE J. Solid-State Circuits*, vol. 43, no. 12, pp. 2939–2957, Dec. 2008.

[3] M. Harwood *et al.*, "A 12.5 Gb/s SerDes in 65 nm CMOS using a baud-rate ADC with digital receiver equalization and clock recovery," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, Feb. 2007, vol. 591, pp. 436–437.

[4] H.-M. Bae *et al.*, "An MLSE receiver for electronic dispersion compensation of OC-192 fiber links," *IEEE J. Solid-State Circuits*, vol. 41, no. 11, pp. 2541–2554, Nov. 2006.

[5] F. Gardner, "Interpolation in digital modems—Part I: Fundamentals," *IEEE Trans. Commun.*, vol. 41, no. 3, pp. 501–507, Mar. 1993.

[6] M. Spurbeck and R. Behrens, "Interpolated timing recovery for hard disk drive read channels," in *Proc. IEEE Int. Conf. Communications*, Jun. 1997, vol. 3, pp. 1618–1624.

[7] K. Mueller and M. Müller, "Timing recovery in digital synchronous data receivers," *IEEE Trans. Commun.*, vol. 24, no. 5, pp. 516–531, May 1976.

[8] R. Johnson, Jr. *et al.*, "Blind equalization using the constant modulus criterion: A review," *Proc. IEEE*, vol. 86, no. 10, pp. 1927–1950, Oct. 1998.

[9] M. van Ierssel *et al.*, "Event-driven modeling of CDR jitter induced by power-supply noise, finite decision-circuit bandwidth, and channel ISI," *IEEE Trans. Circuits Syst. I*, vol. 55, no. 5, pp. 1306–1315, Jun. 2008.

Oleksiy Tyshchenko received the B.A.Sc. degree (with honors) in electrical engineering from the Division of Engineering Science in 2004 and the M.A.Sc. degree in electrical and computer engineering in 2006, both from the University of Toronto, Toronto, ON, Canada. He is currently working toward the Ph.D. degree in the Department of Electrical and Computer Engineering, University of Toronto. In 2006, he spent 6 moths as an intern with Fujitsu Labs of America working on circuit design for high-speed signaling applications.

His research interests are in the design of clock and data recovery (CDR) systems for high-speed ADC-bases receivers. He is also interested in the architecture and circuit design for content-addressable memories (CAM).

Mr. Tyshchenko has held the Natural Sciences and Engineering Research Council of Canada (NSERC) postgraduate scholarship and the Ontario Graduate Scholarship (OGS).

Ali Sheikholeslami (S'98–M'99–SM'02) received the B.Sc. degree from Shiraz University, Shiraz, Iran, in 1990 and the M.A.Sc. and Ph.D. degrees from the University of Toronto, Toronto, ON, Canada, in 1994 and 1999, respectively, all in electrical and computer engineering.

In 1999, he joined the Department of Electrical and Computer Engineering, University of Toronto, where he is currently an Associate Professor. His research interests are in the areas of analog and digital integrated circuits, high-speed signaling, and VLSI memory design. He currently supervises two active research groups in the areas of high-speed signaling and VLSI memories. He has collaborated with industry on various VLSI design research in the past few years, including work with Nortel and Mosaid, Canada, and with Fujitsu Labs of Japan and America. He spent his 2005–2006 research sabbatical year with Fujitsu Labs of Japan and Fujitsu Labs of America.

Dr. Sheikholeslami served on the Memory Subcommittee of the IEEE International Solid-State Circuits Conference (ISSCC) from 2001 to 2004, and on the Technology Directions Subcommittee of the same conference from 2002 to 2005. He currently serves on the Wireline Subcommittee of ISSCC. He presented a tutorial on ferroelectric memory design at ISSCC 2002 and a tutorial on high-speed signaling at ISSCC 2008. He is an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS I: REGULAR PAPERS. He was the program chair for the 34th IEEE International Symposium on Multiple-Valued Logic (ISMVL 2004) held in Toronto, Canada. He is a Registered Professional Engineer in the province of Ontario, Canada.

Dr. Sheikholeslami has received the Best Professor of the Year Award four times since 2000 by the popular vote of the undergraduate students in the Department of Electrical and Computer Engineering, University of Toronto. In 2006, he received the Early Career Teaching Award in recognition of his "superb accomplishment in teaching" from the Faculty of Applied Science and Engineering at the University of Toronto.

Hirotaka Tamura received the B.S., M.S., and Ph.D. degrees in electronic engineering from Tokyo University, Tokyo, Japan, in 1977, 1979, and 1982, respectively.

In 1982, he joined Fujitsu Laboratories, Ltd., Kawasaki, Japan, where he was engaged in research on Josephson devices and other exploratory devices. In 1995, he moved into the area of CMOS circuit design. After working on multi-gigabit DRAMs and ferroelectric nonvolatile memories, he got involved in CMOS high-speed signaling. His current interest covers the circuit topology and architecture of high-speed CMOS interfaces.

Masaya Kibune was born in Kanagawa, Japan, in 1973. He received the B.S. and M.S. degrees in applied physics from Tokyo University in 1996and 1998 respectively.

In 1998, he joined Fujitsu Laboratories, Ltd., Kanagawa, Japan. He has been engaged in research and design of high-speed IO with CMOS.

Hisakatsu Yamaguchi graduated from electrical engineering, Tokyo University of Science, Chiba, Japan in 1994 and received the M.S. degree in electronics engineering from University of Tokyo, Tokyo, Japan in 1996.

In 1996, he joined Fujitsu Labratory, Kawasaki, Japan, where he engaged in research on DRAM with high-speed IF and developed MPEG4 Codec LSI. He is working on developing High-speed IF macro.

Junji Ogawa (M'89) received the B.S. degree in physical chemistry and in applied mathematics from the University of Tokyo, Tokyo, Japan, in 1979 and 1981, respectively.

He joined Fujitsu Ltd., Kawasaki, Japan, in 1981. He was engaged in research and development of DRAM technology in both Fujitsu Ltd. and Fujitsu Laboratories Ltd. His early research included a study of application-specific memory design. He also moved to Fujitsu Laboratories of America Inc. in 1998 and returned to Japan in 2004. He has been working on CMOS high speed interconnection circuits.

Mr. Ogawa is a member of the Information Processing Society of Japan.