A 0.45 pJ/bit, 6.4 Gb/s Forwarded-Clock Receiver With DLL-Based Self-Tracking Loop for Unmatched Memory Interfaces

Soyeong Shin, Student Member, IEEE, Han-Gon Ko, Student Member, IEEE, Chan-Ho Kye, Student Member, IEEE, Sang-Yoon Lee, Student Member, IEEE, Jaekwang Yun, Doobock Lee, Jae-Kang Jung, Suhwan Kim, Senior Member, IEEE, and Deog-Kyoon Jeong, Fellow, IEEE

Abstract—This brief presents a power- and area-efficient forwarded-clock (FC) receiver with a delay-locked loop (DLL)-based self-tracking loop for unmatched memory interfaces. In the proposed FC receiver, the self-tracking loop is composed of two-stage cascaded DLLs to support a burst mode. The proposed scheme compensates for a delay drift neither by relying on data (DQ) transitions nor by re-training but with a write training of the memory controller to fine-tune a data strobe (DQS) path delay through DLLs. The proposed FC receiver is fabricated in the 65-nm CMOS technology and the active area including 4 DQ lanes is 0.0329 mm². After the write training is completed at supply voltage of 1 V, the measured timing margin remains larger than 0.31 UI when the supply voltage drifts in the range of 0.94 V and 1.06 V from the training voltage, 1 V. At the data rate of 6.4 Gb/s, the proposed FC receiver achieves an energy efficiency of 0.45 pJ/bit.

Index Terms—Delay-locked loop, forwarded-clock receiver, memory interface, timing margin, unmatched type receiver, write training.

I. INTRODUCTION

THE high-speed memory interfaces traditionally use source-synchronous architecture in which a transmitter sends data along with a clock and a receiver latches data with the transmitted clock [1]. If the clock is center-aligned with the data and the path delays of clock and data are the same, the receiver has the largest sampling time margin. As shown in Fig. 1(a), the matched type receiver has a tDQS replica delay cell in the DQ path to match the delay of the DQS path. Since the amount of delay variation caused by a voltage or temperature (VT) drift is the same in each path, the VT drift does not degrade the timing margin in the matched type receiver. However, the preferred receiver design of memory interfaces is moving from a matched type to an unmatched type for lower power consumption as the data rate increases. The unmatched type receiver eliminates the tDQS replica cell in the DQ path as shown in Fig. 1(b) to reduce the power consumption. Therefore, it has a different delay between the DQ path and the DQS path. Thus, a memory interface, which adopts the unmatched type receiver, performs a write training to locate a DQS transition on the DQ center at the DQ sampler for the optimal timing margin [2]. In the write training, after a DQS path delay is measured, a memory controller trims DQ later than DQS by the measured DQS path delay, tDQS2DQ. However, the VT drift occurring after the write training changes tDQS2DQ and results in the reduced timing margin because the sampling point deviates from the trained sampling point, the DQ center. The reduced timing margin is a critical problem since a DQ eye width is decreased as the data rate increases. Therefore, compensation for the delay drift is required to maintain the timing margin.

To detect the DQS path delay drift, a periodic incremental training [3] and an internal DQS clock-tree oscillator [4] are suggested. In [3], the delay variation is monitored by measuring the shift of the DQ eye edge in each refresh cycle. During the refresh, the memory controller transmits ‘1010’ pattern to DQ and errors are counted by sweeping the DQS training [3] and an internal DQS clock-tree oscillator [4] are suggested. In [3], the delay variation is monitored by measuring the shift of the DQ eye edge in each refresh cycle. During the refresh, the memory controller transmits ‘1010’ pattern to DQ and errors are counted by sweeping the DQS training changes tDQS2DQ and results in the reduced timing margin because the sampling point deviates from the trained sampling point, the DQ center. The reduced timing margin is a critical problem since a DQ eye width is decreased as the data rate increases. Therefore, compensation for the delay drift is required to maintain the timing margin.

Manuscript received July 31, 2019; revised October 17, 2019; accepted November 26, 2019. Date of publication December 2, 2019; date of current version October 5, 2020. This work was supported by SK Hynix. This brief was recommended by Associate Editor I. Taniguchi.† (Corresponding author: Deog-Kyoon Jeong.)

S. Shin, H.-G. Ko, C.-H. Kye, S.-Y. Lee, J. Yun, S. Kim, and D.-K. Jeong are with the Department of Electrical and Computer Engineering and the Inter-University Semiconductor Research Center, Seoul National University, Seoul 08826, South Korea (e-mail: sshin@isd1.snu.ac.kr; dkjeong@snu.ac.kr).

D. Lee and H.-K. Jung are with DRAM Interface Design, SK Hynix, Icheon 17336, South Korea. Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSII.2019.2957042
Then, the entire process is repeated after the write training to observe the drift of the DQS path delay by comparing the counter values. If a large delay drift is detected by using these methods, the memory controller is required to perform a re-training to change the delay setting, which indicates the relative timing relationship between DQ and DQS from the transmitter. Consequently, the memory controller is responsible for the delay drift compensation which incurs significant design complexity.

Instead of implementing a re-training, embedding a DLL in the DQS path can be a feasible solution for self-tracking because the DLL can fix $t_{DQS2DQ}$ by controlling the delay lines even if the VT drift occurs. Moreover, in the write training, the relative delay between DQ and DQS is set to $t_{DQS2DQ}$. As a result, the DQS transition is always centered at the DQ eye and the timing margin is not decreased by the drift. In this brief, a power- and area-efficient FC receiver with a DLL-based self-tracking loop is presented, which exploits the write training and does not require re-training or DQ transitions for the delay drift compensation.

This brief is organized as follows. Section II details the architecture and the operation of the proposed FC receiver with a DLL-based self-tracking loop. Section III shows the measurement results of the proposed scheme and Section IV concludes this brief.

II. PROPOSED FORWARDED CLOCK RECEIVER

A. Architecture

Fig. 2 shows the overall architecture of the proposed FC receiver. The FC receiver consists of a digital loop filter (DLF), 1:4 deserializers (DES), a divider, DQ samplers, and two-stage cascaded DLLs which include phase detectors (PD) and digitally-controlled delay lines (DCDL). Since DQS inputs ($DQS_t$ and $DQS_c$) have a low voltage swing from 0 V to 0.4 V, a PMOS input strong-arm latch based sampler is used as the PD of the first-stage DLL in Fig. 3(a). On the other hand, as shown in Fig. 3(b), a D-flip-flop is employed as the PD of the second-stage DLL since $DQS_edge_t$ and $DQS_I$ are full swing signals. The DQS path includes a two-stage amplifier, two DCDLs, an I-Q divider, and CMOS buffers. The two-stage amplifier amplifies the DQS inputs from the 0.4-V peak-to-peak swing to the full rail-to-rail swing. As shown in Fig. 4, DCDL is composed of dual coarse delay lines and a phase interpolator (PI) based on tri-state inverters [5] to cover a wide delay range with fine resolution while avoiding a boundary switching problem. In the coarse delay line, CLKU and CLKD always have a 2*$t_NAND$ time difference. The 6-bit thermometer codes $CU$ and $CD$ control the number of on-state NANDs in CLKU and CLKD paths, respectively, to adjust the delay. The PI interpolates CLKU and CLKD by adjusting the number of the on-state tri-state inverters in each path. The length of the DCDL codes is determined by the required delay range and the DCDL has an effective resolution, $2^*$NAND/15. The I-Q divider generates 4-phase clocks for a quarter-rate clocking. Both DCDL1 and DCDL2 are adjusted to fix the total DQS path delay as $N$ times the unit interval (UI) by two-stage cascaded DLLs.

B. Operation

Fig. 5 shows the timing diagram of the proposed scheme. In Fig. 5(a), $DQS_I$ is $N$*UI delayed from the DQS inputs by means of cascaded DLLs. Since the optimal sampling point is set in the write training when $t_{DQS2DQ}$ is $N$*UI and the cascaded DLLs always fix the DQS path delay to $N$*UI by adjusting DCDLs, the sampling point is not changed even if the VT drifts. Therefore, the timing margin, which is defined as the minimum value of the left margin and the right margin from the DQ center, can be kept constant by the DLLs even though the VT drift occurs. As shown in Fig. 5(b), in the burst mode and long $t_{DQS2DQ}$ condition, phase comparison between the DQS inputs and $DQS_I$ cannot be carried out because the DQS inputs do not toggle when $DQS_I$ transition occurs. Thus, to support the burst mode, $DQS_edge_t$ and $DQS_edge_c$ are generated for the phase comparison in the first-stage DLL. In other words, the DLLs are separated...
Fig. 6. Lock point setting method (a) 1st stage DLL and (b) 2nd stage DLL.

Fig. 7. A flow chart of the digital loop filter operation.

Fig. 8. Chip microphotograph and core layout of the proposed scheme.

The first-stage DLL makes DQS_edge pair (DQS_edge_t and DQS_edge_c) and the second-stage DLL generates the DQ sampling signal, which is \( N \times UI \) delayed from the DQS inputs in the locked state.

The first-stage DLL creates DQS_edge pair, which are \( N_1 \times UI \) delayed version of the DQS inputs, by edge-aligning DQS_edge pair with the DQS inputs by controlling DCDL1. Since DQS_edge pair are produced by delaying the DQS inputs, the amount of the delay should be minimized for low jitter condition in the locked state. Thus, the first-stage DLL aligns the rising transition of DQS_edge_t with the transition edge of DQS_c which is closest to the rising transition of DQS_edge_t but is later than the rising transition of DQS_edge_t when the codes of the coarse delay line of DCDL1 are at the minimum. Fig. 6(a) shows the lock point setting method of the first-stage DLL. Through phase comparison of DQS_edge_t and DQS_c, PD_L1 calculates the lock point of the first-stage DLL under the minimum coarse delay condition of DCDL1. The settled lock point is the nearest transition edge of DQS_c that can be aligned with the rising edge of DQS_edge_t by the delay increment of DCDL1. For instance, if PD_L1 is '0', the rising edge of DQS_edge_t leads the rising edge of DQS_c and it should be aligned with the rising edge of DQS_c by increasing the delay of DCDL1. Otherwise, if PD_L1 is '1', the rising edge of DQS_edge_t should be aligned with the falling edge of DQS_c. The final lock point can be varied with the initial delay condition or PVT variations due to the different positions of the rising edge of DQS_edge_t. Similarly, the lock point of the second-stage DLL is determined as the rising transition of DQS_edge_pair which is closest to the rising transition of DQS_I when the codes of the coarse delay line of DCDL2 are at the minimum for the same reason. The rising edge of DQS_I can be aligned with the determined lock point by increasing the delay of DCDL2. The combination of PD outputs, PD_t and PD_c, indicates the position of the rising transition of DQS_I. As shown in Fig. 6(b), in the case shown on the left, the rising edge of DQS_I is in between the rising edges of DQS_edge_c and DQS_edge_t, shaded in blue. Thus, the delay of DCDL2 should be increased to make the rising edge of DQS_I aligned with the rising edge of DQS_edge_t. In the case shown on the right, the rising edge of DQS_I can be aligned with the rising edge of DQS_edge_c by increasing the delay of DCDL2. As a result, the DQS path delay from the DQS inputs to DQS_I is always \( N \times UI \) in the locked state. If the duty cycle of the DQS inputs is not 50%, the DQS path delay will not be the same as \( N \times UI \). However, only fixing the DQS path delay is important since the fixed sampling point will be at the optimum when the memory controller performs the write training under this condition.

Fig. 7 shows a flow chart of the DLF operation. A coarse sweep mode and an update gain of each DCDL can be selected and controlled respectively by I^2C for adjusting the locking time of each DLL. The combination of PD outputs, PD_t and PD_c, indicates the position of the rising transition of DQS_I. As shown in Fig. 6(b), in the case shown on the left, the rising edge of DQS_I is in between the rising edges of DQS_edge_c and DQS_edge_t, shaded in blue. Thus, the delay of DCDL2 should be increased to make the rising edge of DQS_I aligned with the rising edge of DQS_edge_t. In the case shown on the right, the rising edge of DQS_I can be aligned with the rising edge of DQS_edge_c by increasing the delay of DCDL2. As a result, the DQS path delay from the DQS inputs to DQS_I is always \( N \times UI \) in the locked state. If the duty cycle of the DQS inputs is not 50%, the DQS path delay will not be the same as \( N \times UI \). However, only fixing the DQS path delay is important since the fixed sampling point will be at the optimum when the memory controller performs the write training under this condition.
DCDL are increased until the PD output pattern changes from the initial value which is calculated at the lock point setting step. For example, if the PD_L1 is ‘0’ in setting lock point when the coarse delay of DCDL1 is at the minimum, the PD_L1 can be changed to ‘1’ by increasing the coarse delay of DCDL1 which implies the rising edge of DQS_edge_t lags behind the rising edge of DQS_c. This means the coarse sweep of the first-stage DLL is done because the rising edge of DQS_edge_t is later than the lock point, the rising edge of DQS_c. The coarse sweep of the second-stage DLL is done in the similar way as the first-stage DLL. Then, the codes of DCDL1 and DCDL2 are updated simultaneously corresponding to the PD outputs and the determined lock points as shown in Fig. 6. The lock point determines which edge to lock and PD outputs indicate the current position of the rising edge of signals to be aligned. In Fig. 6(a), if the lock point of the first-stage DLL is the rising edge of DQS_c and PD_L1 indicates that the rising edge of DQS_edge_t is in the UP region, the code of DCDL1 is increased to align the rising edge of DQS_edge_t to the lock point. Otherwise, if PD_L1 is ‘1’ and the rising edge of DQS_edge_t is in the DN region, the code of DCDL1 is decreased. Also, the code of DCDL2 is controlled by the location of the DQS_I rising transition and the lock point of the second-stage DLL as shown in Fig. 6(b).

### Table I

**Measured Power Breakdown**

<table>
<thead>
<tr>
<th>Block</th>
<th>Power</th>
<th>Block</th>
<th>Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCDL</td>
<td>3.35 mW</td>
<td>DQS path</td>
<td>3.05 mW</td>
</tr>
<tr>
<td>Sampler</td>
<td>3.14 mW</td>
<td>DES</td>
<td>0.82 mW</td>
</tr>
<tr>
<td>DLF</td>
<td>0.67 mW</td>
<td>PD</td>
<td>0.42 mW</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>11.45 mW</strong></td>
<td><strong>at 6.4 Gb/s</strong></td>
<td><strong>1-V supply, 4 DQ</strong></td>
</tr>
</tbody>
</table>

The proposed FC receiver is fabricated in the 65-nm CMOS technology. The chip microphotograph and the core layout of the FC receiver are shown in Fig. 8. The DCDL in a transmitter and the emulated channel are implemented to behave in place of the memory controller and the memory channel, respectively [6]. Using standard cells, the DLF is fully synthesized and automatically placed and routed. The active area of the proposed FC receiver is 0.0329 mm², which includes 4 DQ lanes.

### III. Measurement Results

The graph of timing margin versus supply voltage is shown in Fig. 9. The DLL-off case is explored when all DCDL codes of receiver domain are set at the minimum. At 1 V in the DLL-on case, the timing margin is reduced from 0.36 UI to 0.33 UI due to the increased DQS path delay compared with the DLL-off case. However, the timing margin in the DLL-on case remains larger than 0.31 UI while the supply voltage drifts in the range of 0.94 V and 1.06 V after the write training is done at 1 V.

Fig. 10 shows the measured DQ eye diagram and the bathtub curve at three different supply voltages when the write training is carried out at 1 V. Contrary to the DLL-off case, the DQ eye does not shift because the sampling point is not modified in the DLL-on case.
TABLE II

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture</td>
<td>Digital DLL</td>
<td>Digital DLL</td>
<td>PI</td>
<td>Analog DLL</td>
<td>Analog PD+</td>
<td>Digital DLL</td>
<td></td>
</tr>
<tr>
<td>Data rate (Gb/s)</td>
<td>8</td>
<td>6.4</td>
<td>14</td>
<td>10</td>
<td>12.5</td>
<td>4.8</td>
<td>6.4</td>
</tr>
<tr>
<td>Clock rate (GHz)</td>
<td>2</td>
<td>3.2</td>
<td>3.5</td>
<td>5</td>
<td>6.25</td>
<td>1.2</td>
<td>1.8</td>
</tr>
<tr>
<td>Power (mW)</td>
<td>10.4*</td>
<td>6.71*</td>
<td>7.84</td>
<td>7.1</td>
<td>4.5</td>
<td>7.04**</td>
<td>11.45**</td>
</tr>
<tr>
<td>VDD (V)</td>
<td>1.25</td>
<td>0.85</td>
<td>0.8</td>
<td>1</td>
<td>0.9</td>
<td>1.1</td>
<td>1</td>
</tr>
<tr>
<td>Power efficiency (pJ/b)</td>
<td>1.3</td>
<td>1.05</td>
<td>0.56</td>
<td>0.71</td>
<td>0.36</td>
<td>0.37</td>
<td>0.45</td>
</tr>
<tr>
<td>Area (mm²)**</td>
<td>0.225</td>
<td>0.02</td>
<td>0.36</td>
<td>0.014</td>
<td>0.025</td>
<td>0.0056</td>
<td>0.008</td>
</tr>
<tr>
<td>Require data transition ?</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

*Excludes equalizer power  **includes 4 data lanes  ***area per lane

The simulated tracking behavior of the proposed FC receiver (a) when voltage drift occurs and (b) when temperature drift occurs.

The simulated tracking behavior of the proposed scheme when the 0.06 V supply voltage drift or 60 °C temperature drift occurs from 200 ns to 300 ns is shown in Fig. 11. When the DCDL update gain is high, the sampling time difference from the settled point remains constant due to a high loop bandwidth.

Table I shows the measured power breakdown of the proposed FC receiver. The total power consumption is 11.45 mW from 1-V supply at 6.4 Gb/s including 4 DQ lanes. In Table II, the performance of the proposed FC receiver is summarized and compared with those of the state-of-the-art FC receivers [6]–[11]. The proposed FC receiver uses a small area and low power. Furthermore, the FC receiver is able to operate in the absence of data transitions since the sampling point is fixed by the DLL using only DQs and the write training sets the sampling point as the optimal sampling point for the DQ eye centering.

IV. CONCLUSION

A small-area and power-efficient FC receiver with DLL-based self-tracking loop for unmatched memory interfaces is proposed and implemented in the 65-nm CMOS technology. The proposed FC receiver adopts a cascaded DLL architecture to support the burst mode. The proposed scheme compensates for the VT drift by fixing IDQS2DQ using a DLL. Therefore, it does not require a re-training in the memory controller. Since it utilizes the write training, it does not need DQ transitions. In addition, it does not increase the capacitance at DQ pins because monitoring DQ is not necessary for VT drift compensation. The proposed FC receiver achieves the timing margin larger than 0.31 UI with power efficiency of 0.45 pJ/bit at 6.4 Gb/s while the supply voltage drifts in the range 0.94 V and 1.06 V.

REFERENCES