A Wide-Range Mixed-Mode DLL for a Combination 512 Mb 2.0 Gb/s/pin GDDR3 and 2.5 Gb/s/pin GDDR4 SDRAM

Feng Lin, *Member, IEEE*, Roman A. Royer, Brian Johnson, *Member, IEEE*, and Brent Keeth, *Member, IEEE*

***Abstract*—A mixed-mode delay-locked loop (MDLL) for a 512 Mb graphics SDRAM is presented in this paper. The MDLL extends its lock range into the gigahertz realm by applying clock division and analog phase generation (APG). The divided clock from the MDLL is used for clocking logic and tracking deterministic access latency in the SDRAM. A short discussion of some of the side effects and advantages of using a divided, multi-phase clock for logic operation is presented. A low-power clock distribution network (CDN) based on the presented MDLL is also disclosed. Fabricated in a 1.5 V 95 nm triple-metal CMOS process, the MDLL achieves a measured RMS jitter of 4.6 ps and peak-to-peak jitter of 38 ps at GDDR4 mode with a 1 GHz clock. Power consumption for the entire MDLL-based CDN is 107 mW at 800 MHz and 1.5 V.**

***Index Terms*—Delay-locked loops (DLL), clock distribution network (CDN), analog phase generation (APG), de-skewing, duty-cycle correction (DCC), synchronous DRAM (SDRAM).**

I. Introduction

As THE CLOCK SPEED for I/O interfaces pushes well into the gigahertz range, designing timing circuits, such as de-skewing circuitry and clock distribution networks (CDNs), becomes a challenge. The on-die synchronization circuit must cover a wide operation range and, at the same time, achieve good jitter performance. For a combination GDDR3/GDDR4 memory interface, the minimum and maximum clock frequency is 400 MHz and 1.25 GHz, respectively, while the typical operating supply voltage ranges from 1.5 V to 1.8 V. Inclusion of on-die clock synchronization is necessary for control of output timing skew between the system clock and the output data from the DRAM. However, there are side effects from the use of synchronization circuits when the system clock period becomes comparable with the I/O delay of the DRAM. Variations in process, voltage, and temperature (PVT) can easily cross cycle boundaries and make read latency prediction difficult. A phase-tolerant latency tracking scheme can be found in [7]. Besides latency control, high-speed clock distribution is also susceptible to duty-cycle distortion, power supply noise, and timing mismatch. Power and jitter for timing circuits must be carefully gauged to achieve overall better performance.

Traditionally, a delay-locked loop (DLL) is selected for clock synchronization due to its close-loop architecture and relatively simple implementations in both digital [1], [2] and analog [3], [4] domains. To achieve fast and tight locking, a combination of both digital and analog approaches has been reported in [5], [6], but the maximum clock frequency is below 220 MHz. To reduce duty-cycle distortion for a double data rate (DDR) memory system, a symmetrical delay cell [1] is found useful when constructing a digital delay line. A digital duty-cycle correction (DCC) [2] provides a close-loop solution with extra hardware and power consumption. Analog approaches can also be found in [3] and [4] with multiphase generation and dedicated DCC circuitry.

In this paper, a mixed-mode DLL (MDLL) with clock divider and analog phase generator (APG) is proposed for wide-range high-speed clock synchronization. Based on the operating mode (GDDR3/4), the clock divider, like in [2], can slow down the internal operation by utilizing half-speed clocks. Instead of using an extra digital DLL for DCC [2], a multiple-phase analog DLL is applied for edge recovery and built-in duty-cycle correction. A portion of this discussion will also focus on the side effects that the clock synchronization circuit choices have on the command decoder implementation and the read data output timing path. It will also be shown that power and area can be saved with the proposed MDLL, and the measured maximum achievable clock frequency is 2 GHz at 2.12 V supply.

Following the introduction, Section II discusses overall clocking architecture for the proposed MDLL-based CDN and the effects of this architecture on the peripheral DRAM logic circuits. Section III describes the digital DLL and the proposed APG using a modified voltage-controlled delay line (VCDL) and dual-edge phase detector to achieve multiphase generation and built-in DCC. Section IV shows experimental results and Section V outlines conclusions.

II. Clocking Architecture and Effects on DRAM Logic Implementation

The proposed mixed-mode DLL (MDLL) is shown in Fig. 1. A conventional digital DLL (like the one used in [5]) with fast-lock and fine resolution (via phase interpolation) is adopted for clock synchronization and de-skewing. A clock divide-by-2 circuit is inserted into the input path to slow down the internal clock speed for GDDR4 mode of operation. A MUX is either used to select full-speed or half-speed clock. The clock divider effectively doubles the I/O operating speed with minimum impact on internal timing circuits. When half-speed clock is used for power saving and better signal integrity for high-speed clock

Manuscript received September 4, 2007; revised December 13, 2007.

The authors are with Micron Technology, Boise, ID 83707-0006 USA (e-mail: flin@micron.com).

Digital Object Identifier 10.1109/JSSC.2007.916623

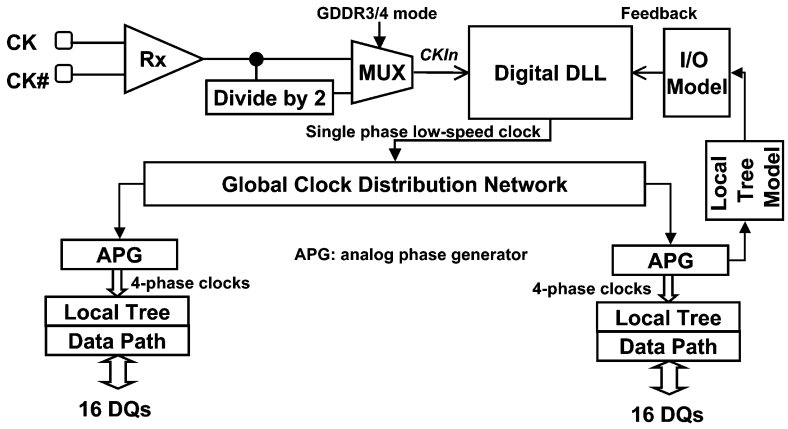


Fig. 1. Mixed-mode DLL used for clock distribution and multiphase generation.

distribution, an edge recovery circuit is required for a DDR interface, where falling edges of the clock are also used for data operations. An analog DLL using a VCDL is picked for the edge recovery. During power-up and initialization, the analog phase generator (APG) is locked first, followed by the digital DLL synchronization. During digital synchronization, the phase detector of the APG is disabled for a short period of time when the output of the DLL is switched for quick initialization (more details follow). A detailed implementation of the digital DLL and multiphase generator will be discussed in Section III.

The MDLL-based clock distribution network (CDN) can be grouped into two parts: global clock tree and local clock tree (as shown in Fig. 1). With 32 DQs placed in four quadrants of the die for graphic DRAM, clock trees can consume significant power. Matching is another concern when distributing multiphase clocks. Instead, in this work, only a single-phase clock out of the digital DLL is distributed globally over a long distance. Two analog phase generators (APG) are placed at each end of the global tree to generate multiphase clocks. The outputs of the APG will drive the local clock tree, which eventually clocks the output data path. For low-speed operation, a clock divider can be used instead to generate 4-phase clocks to save power and lock time. A 4-phase clocking diagram is illustrated in Fig. 2. CK is the external clock running at full-speed, where $C0$, $C90$, $C180$, and $C270$ are internally generated 90-degree phase-shifted half-speed clocks. Only the rising edges of the 4-phase clocks are used for clocking the data path. The duty-cycle for each clock phase is not critical as long as each clock distribution path is matched. The multi-phase clock output from the APG circuits affects circuit and clocking methodologies employed in the I/O data path and the command decoder circuits. The advantages of using the divided, multi-phase clocks for command decode and read data are slightly offset by increased logic complexity.

The command decode circuits for a synchronous DRAM require that a portion of the decode circuitry be devoted to tracking synchronous, deterministic timing of both read and write data latency. Latency is measured in command clock cycles (t_{CK}) and programmed into the DRAM mode register usually as part of the power-on/initialization sequence. For example, when a read command is received by the DRAM, the command decoder will trigger an immediate array access and also generate

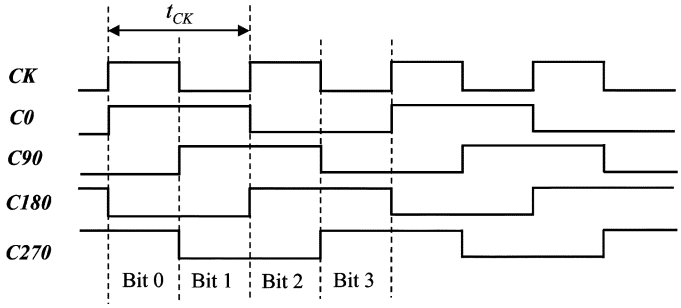


Fig. 2. Timing diagram for 4-phase clocking.

a read signal synchronized to the internal command clock. If the internal command clock is frequency divided then, because commands are aligned only to positive edges of the full-frequency system clock, there are two possible phase alignments of the synchronously generated, internal read signal. Fig. 3 shows how this alignment occurs for DDR4 operation. Quite simply, the synchronous read signal will align with either the $C0$ clock or the $C180$ clock phase because these two clock phases correspond with positive transitions of the system clock, and, hence, DRAM commands.

Dynamic logic and skew tolerant techniques [11] can be employed [11] for the command decoder and timing control. This style of logic is made more feasible by a multi-phase clocking environment. A secondary DLL or PLL is be used to align a multi-phase clock to the delay-matched command/address capture clock and write strobe (WS) distribution paths [7]. The outputs of the capture latches are converted to dual-rail, monotonically rising signals with a pulsewidth of two CK periods as shown in Fig. 4. Because DRAM commands are generally separated by at least 2 CK periods, direct use of the divided clock is acceptable. There are, however, exceptions for ACTIVATE and PRECHARGE commands. If the DRAM protocol does not allow for a divided clock, this method can still be implemented by artificially generating the quadrature phases using gate delays that approximate the quadrature shift for the highest operating frequency.

Self-timed decode and control paths are formed by generating monotonic signals from dynamic logic gates. Translating

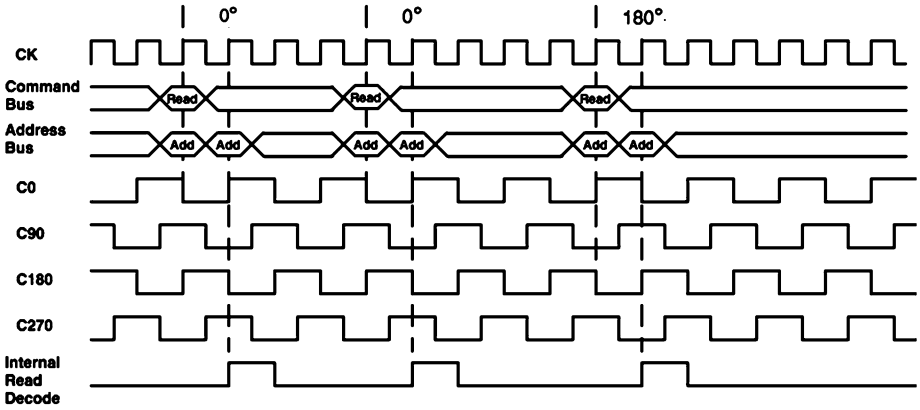


Fig. 3. Multi-phase decoder output.

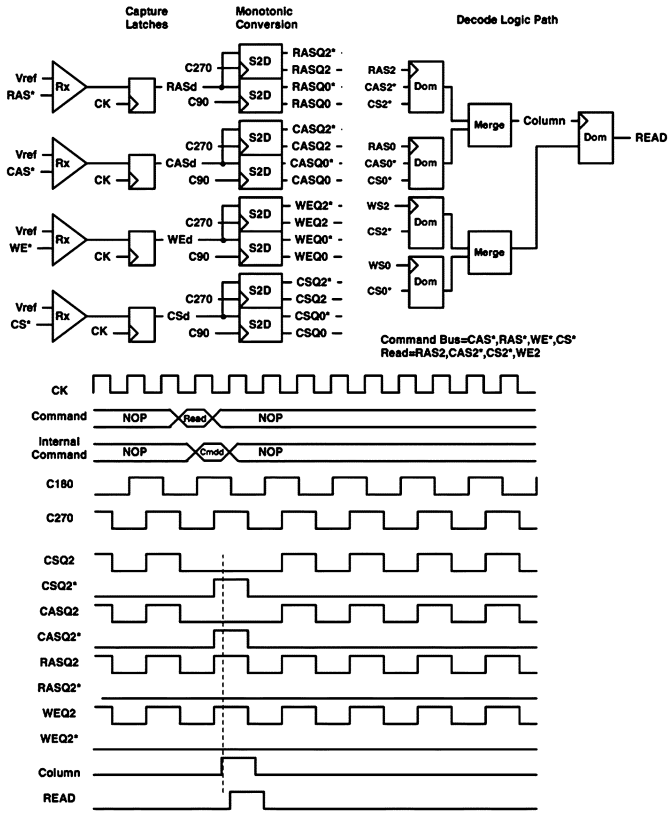


Fig. 4. Dynamic command decoder.

the static outputs of the command capture latches to monotonically rising signals avoids static hazards because high-to-low transitions are not evaluated at the inputs to the dynamic logic gates—only low-to-high transitions can occur just prior to or during the evaluation phase of the dynamic gates. These logic paths are sequenced by the multi-phase clock but are not directly clocked by the output of the APG beyond the first stages of logic. The advantage of using this type of logic is that the command decoder delay is determined by the gate delay between levels of decode. More importantly, the output of the command decoder is hazard free without having to resort to a realignment register with the associated timing overhead for such a topology. Fig. 4 is an example of such a decode path for a read command. In Fig. 4, following the capture latches, the first stage of decode is a circuit used for converting the input control signals into monotonically rising signals [12]. The conversion circuit (S2D) is clocked by the (quadrature) clocks $C90$ and $C270$, which are shifted by a quarter-period relative to the $C0$ and $C180$ clocks. Ideally, phases $C0$ and $C180$ align with the output of the capture latches. When the clock to data out (t_{CO}) for the command capture latches is significant, a delay model for the capture latch circuit is placed in the input path of the reference clock of the secondary DLL to center the divided quadrature phases within the output symbol of the capture latches. Alternatively, this conversion function can be directly implemented in the capture latch. For this work, the capture latch was kept separate from the conversion circuits in order to provide static command signals for manufacturing tests. The conversion circuit (Fig. 5) must adhere to the setup and hold requirements of a synchronous storage element (the soft clock edge of the conversion circuit aids setup timing). In Fig. 4, the output of the conversion circuit is dual-rail, meaning the circuit generates a true and complement output (both are monotonic) for functional completeness. The outputs of the conversion circuits then become clocks to the dynamic decoder gates.

Multi-phase clocking also affects input and output timing logic for the DRAM data I/O circuit paths. Deterministic read latency is a requirement for GDDR3/4. Because two internally generated clock phases $(C0, C180)$ represent rising edges on CK, for correct read latency tracking, it is necessary to determine the correct internal clock phase from which to fire the output data before array data enters the output data path. Just as DRAM commands can correspond to either clock $C0$ or $C180$, burst output data in a multi-phase system will also be required to start on output clock phase $C0$ or $C180$ in the read clock domain. A method for tracking deterministic read latency was outlined in [7]. A top-level diagram of the latency tracking circuit blocks from [7] are shown in Fig. 6. Following is a brief outline of the operation of this circuit.

Referring to Fig. 6, upon power-up or reset the block labeled *Initialization FSM* waits for the APG and DLL circuits to achieve lock following which the initialization circuit simultaneously disables the resets for the two Gray code counters. The Gray code counter in the read clock domain is immediately enabled on the following clock edge while the Gray code

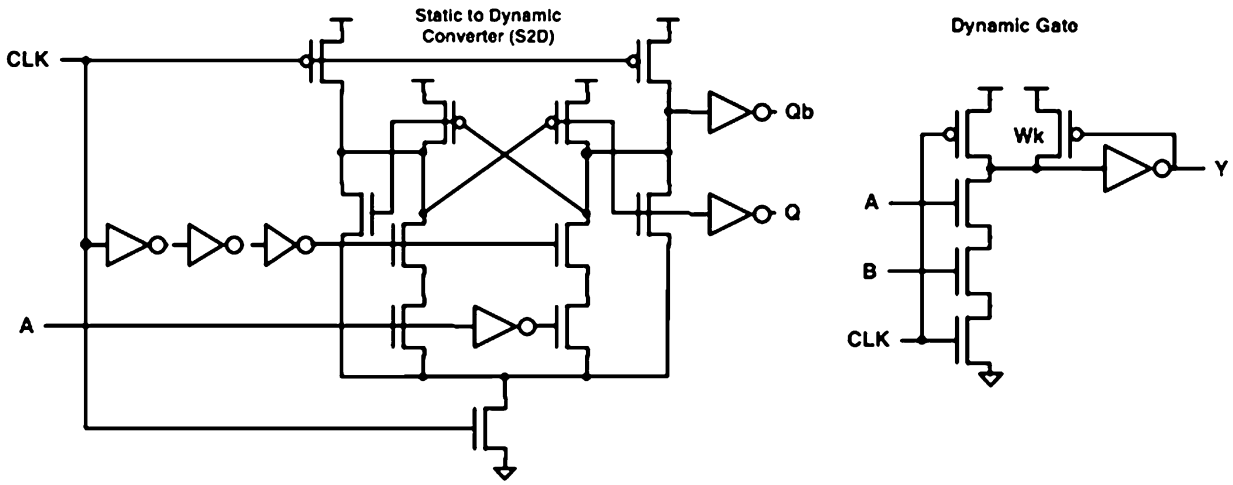


Fig. 5. Command decoder dynamic circuits.

counter in the command/address (C/A) clock domain is enabled following a delay determined by a replica of the DLL feedback I/O timing model. By sending the reset through the replica DLL I/O model from the read clock domain to the C/A clock domain, the reset signal is automatically synchronized to the C/A clock domain because of the phase alignment of the DLL (the C/A clock lags the read clock by the delay of the I/O model). This portion of the initialization results in transitions from the read clock Gray code counter that lead the C/A clock Gray code counter by the I/O model delay. Prior to starting the counters, one of the Gray code counters is preloaded with the programmed read latency compensated by any full clock periods used for synchronization in the read timing logic path (SP). In Fig. 6, the C/A Gray code counter is loaded with the value CL-SP where CL is the programmed read (CAS) latency value expressed as an integer number of system clock (CK) cycles. Under these conditions, any value that occurs at the output of the C/A Gray code counter will occur at the read clock Gray code counter with a delay of $\mathrm{tICL} = (\mathrm{CL} - \mathrm{SP}) - (t_{\mathrm{in}} + t_{\mathrm{out}})$ where t_{in}+t_{out} is the delay through the DLL feedback I/O model. Following initialization, when a read command is received, the value from the C/A Gray code counter is loaded into the FIFO. When the corresponding value is detected from the read clock Gray code counter, the digital comparator signals a 'Start' to the output enable timer and the output data path is enabled for a read burst with the correct latency as referenced to the DRAM system clock.

The multi-phase clocking scheme outlined in this work results in some interesting side effects for the read latency tracking method described above. First, the use of Gray code counters is necessitated by the procedure of loading a count value into the FIFO. By using a Gray code sequence, the probability of a load failure and, especially, a digital comparator miscalculation because of skew between multiple transitioning signals is greatly reduced. Another interesting aspect of using a Gray code counter (reflected binary code [14]) in a divided clock application is that the sequence of the counter is well suited for multi-phase clocking. The Gray code sequence for a 4-bit counter is shown in the truth table inset of Fig. 7 (This figure is a timing diagram, logic diagram and truth table for the gray code counter). Notice that the LSB of the Gray code transitions on opposite cycles from the upper 3 bits. It was established that transitions on clocks $C0$ and $C180$ correspond to positive edge transitions on the DRAM system clock. For a 4-bit Gray code counter, if $C0$ is used to clock the LSB of the counter output and $C180$ is used to clock the upper 3 bits, we get a sequence that corresponds to the timing diagram of Fig. 7. The ability to mux the clock phases for the counter that is preloaded counter must also be considered in order to correctly align the clock phases with the preloaded counter offset (CL-SP). The clock phases may need to switch depending on the calculated preload value. If the preload value corresponds to a transition on the upper 3 bits following initialization, and the initialization phase starts on $C0$ in the read clock domain, then the clocks must be muxed such that $C0$ clocks the upper 3 bits and $C180$ clocks the LSB of the C/A Gray code counter. Finally, clocking the Gray code counter in the fashion described here is essentially a method of doubling the apparent clock frequency at the output of the Gray code counter relative to the frequency of the separate, divided input clocks. Thus, the Gray code output transitions at the full-frequency system clock rate. Another advantage of using a divided clock is that the probability of initialization failure at the receiving flip-flop when starting the C/A Gray code counter for latency tracking is cut in half because the clock period is doubled [13].

Another important detail for read latency tracking when using a frequency divided clock is that there are two possible output clock phases from which the output data path is enabled for the start of a data burst. Consideration must be given to the programmed latency and the clock phase from which the read command is decoded in order to determine which read clock phase will enable the output data path. If the programmed read latency (CL) is 12 and the number of synchronization points (SP) in the logic path is 3 then the resulting discrete portion of *tICL* is equal to 9 ($\mathrm{CL} - \mathrm{SP} = 9$). In this case, the Gray code latency counters are separated by an odd number of system cycles, which results in enabling the output path on the complementary clock phase from which the read command is decoded. For example,

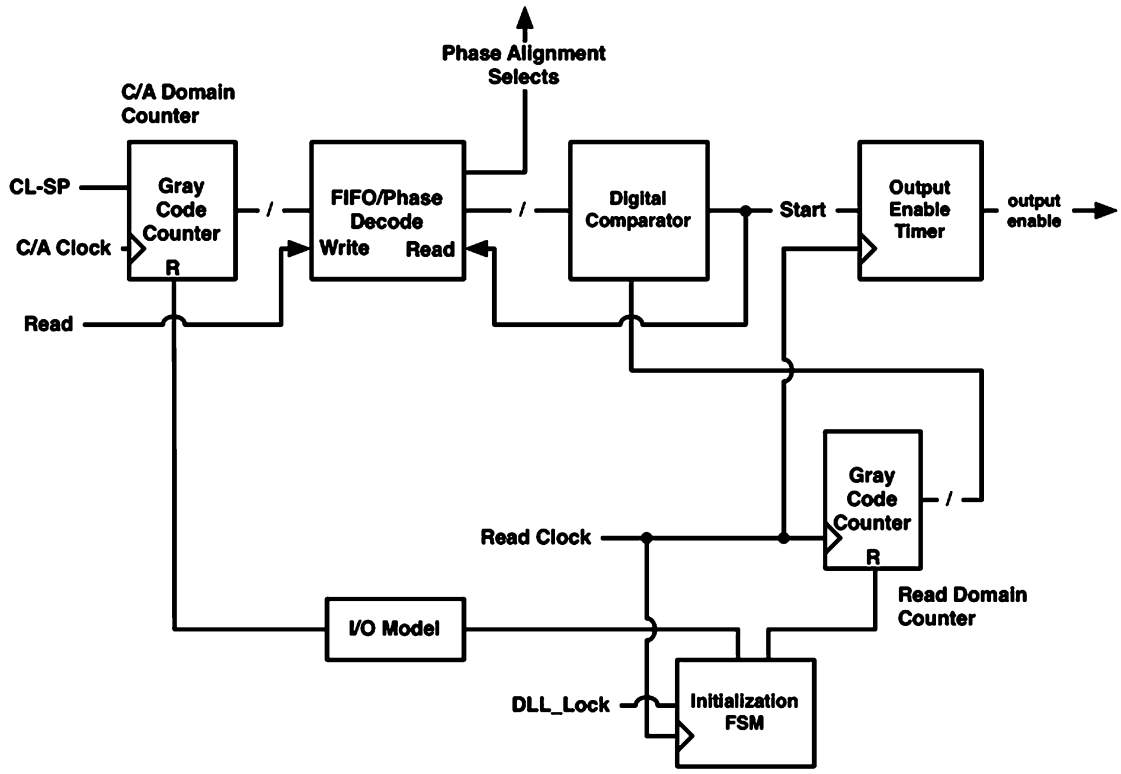


Fig. 6. Read latency tracking circuit.

if the read command is received on $C0$, then counting an odd number of system clock cycles results in the output data path being enabled on $C180$. This information must be forwarded to the output data path prior to array data arrival so that the read data from the memory array can be correctly aligned with the clock phases of the output data clock (phase $C180$ in the stated case). Because of the high data rates and short column cycle times for the GDDR3/4 device, several read commands can be initiated before the read latency expires from the first in a series of open page accesses. For this reason, it becomes necessary to pipeline the phase alignment information from the C/A clock domain to the read clock domain for each read command in order to correctly align the data with the clock.

Fig. 8 is a block diagram detailing a mux for adjusting the data path to align the data with the correct clock cycle for the multiphase output clock. The mux select value is calculated based on the conditions of the programmed latency and Gray code state when the read command is decoded. The mux select values are pipelined to the output data path in a FIFO where each value in the FIFO corresponds to a captured Gray code count value stored in the read latency FIFO of Fig. 6. Each phase alignment value must be forwarded since non-consecutive reads can be in transit before the completion of an array access. The FIFO which presents the mux select is referred to as an "orthogonal" FIFO because it presents the mux select perpendicular to the direction of the array data flow into the output data path. Fig. 8 shows this relationship for clarification.

The mux select can be directly determined from the captured C/A Gray code count value. Again, looking at the truth table in Fig. 7, by calculating the parity of the captured Gray code count the phase of the clock relative to the output clock phase from which the read command was generated can be determined. Further, the parity of the captured Gray code value is combined with the even/odd state of the compensated programmed latency (CL$-$SP) to determine the value of the mux select that is loaded into the orthogonal FIFO and ultimately used for data alignment in the output data path. This calculation tracks transitions on both $C0$ and $C180$ clock phases.

Now that some of the side effects and advantages of using a multi-phase clock in a SDRAM where deterministic latency is a part of the specification have been outlined, the remainder of this work will focus on the implementation of the mixed-mode DLL.

III. Digital DLL and Analog Phase Generator (APG) With Built-in DCC

In order to evaluate the proposed clocking architecture, a comparison between traditional digital DLL (DDLL) and proposed MDLL is summarized in Table I. An 800 MHz external clock is fed into both DLLs. The clock is running at full speed for the DDLL and half speed for the MDLL. Four-phase clocks are generated by a clock divider for the DDLL and an APG for the MDLL, respectively. Simulation is performed at 1.5 V, typical process and 85 C. Duty-cycle correction (DCC) for the DDLL is not included in this simulation, which requires additional power, area, and lock time for the conventional approach. From Table I, the proposed MDLL consumed less power for both DLL and global clock tree (including two APGs) due to

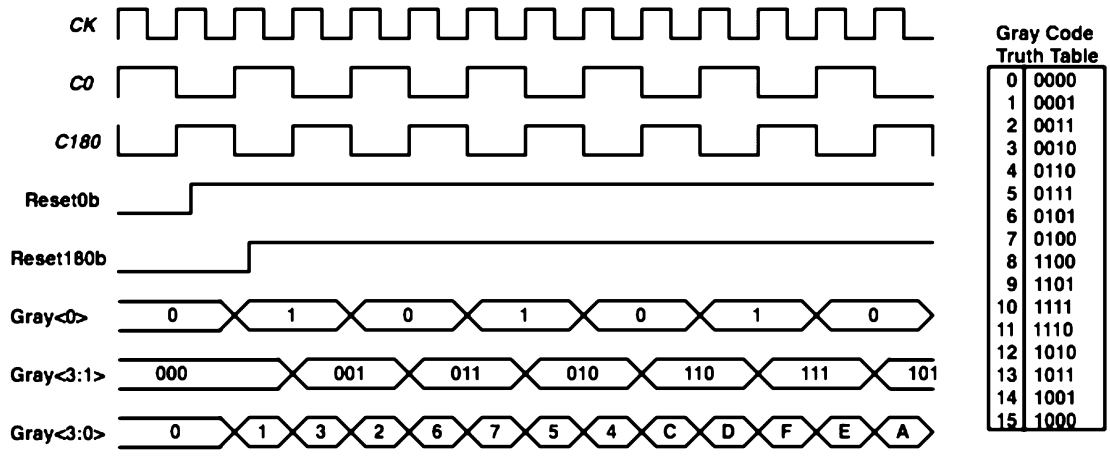


Fig. 7. Multi-phase Gray code counter output.

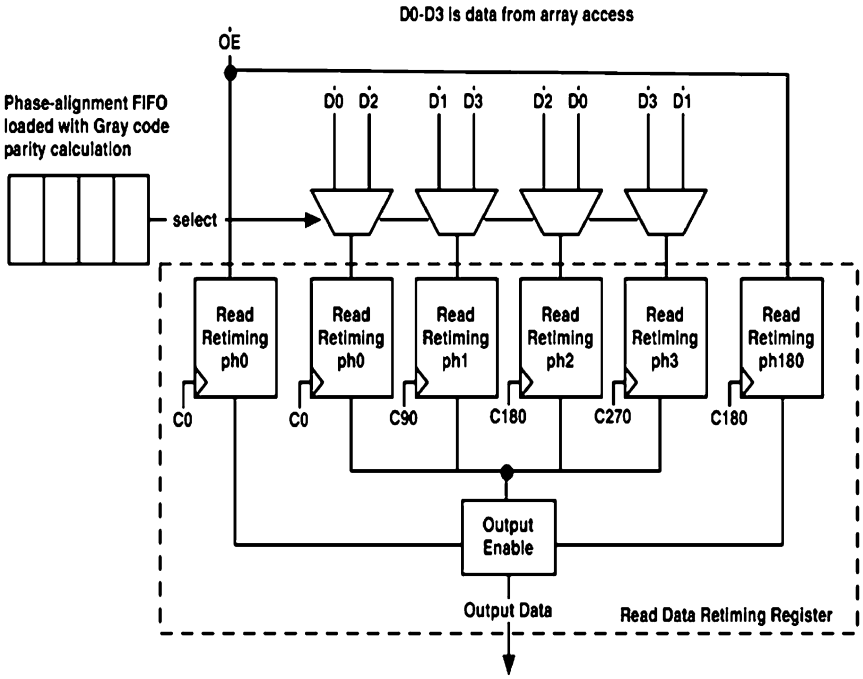


Fig. 8. Multiplexor select FIFO and output data path relationship.

TABLE I
COMPARISON BETWEEN CONVENTIONAL DIGITAL DLL (DDLL) AND PROPOSED MIXED-MODE DLL (MDLL) AT 1.5 V, 85 C, AND 800 MHZ EXTERNAL CLOCK

	DLL current (mA)	Global clock tree + Phase Generator (mA)	Local clock trees (mA)	Lock time (ns)
DDLL	14	13.3	50	56
MDLL	8.2	13.1	50	146

half-speed clocking. However, the lock time did increase for analog phase generation.

The digital DLL used in the MDLL can achieve fast lock with a measure-controlled delay (MCD) line, shown in Fig. 9. The time-to-digital conversion (TDC) and digital-to-time conversion (DTC) will start during measure initialization. The input delay $(t_{\rm in})$ and output delay $(t_{\rm out})$ are modeled with an I/O model $(t_{\rm in}+t_{\rm out})$. Delay for clock distribution is $t_{\rm Tree}$ in Fig. 9. The intrinsic loop delay $(t_{\rm ILD}=t_{\rm Tree}+t_{\rm in}+t_{\rm out})$ is measured by the measure delay array (MDA) and the difference $(N^*t_{\rm CK}-t_{\rm ILD})$ is stored in the forward delay array (FDA). Parameter $t_{\rm CK}$ is the clock period and N is an integer number. When $t_{\rm CK} > t_{\rm ILD}, N=1$.

During initial measurement, the FDA is bypassed by the two-input MUX. A *Start* signal triggered by the rising edge of the internal clock (CKIn) initializes the measurement. As shown

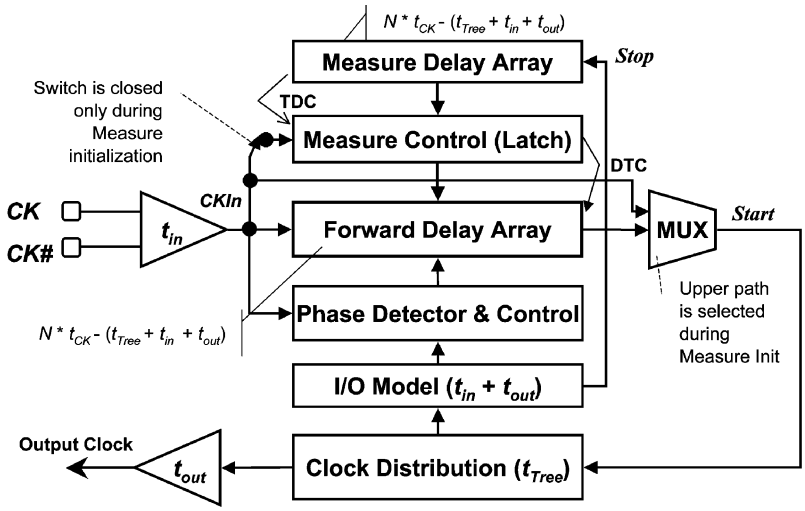


Fig. 9. Close-loop, measure-controlled delay (MCD) based digital DLL.

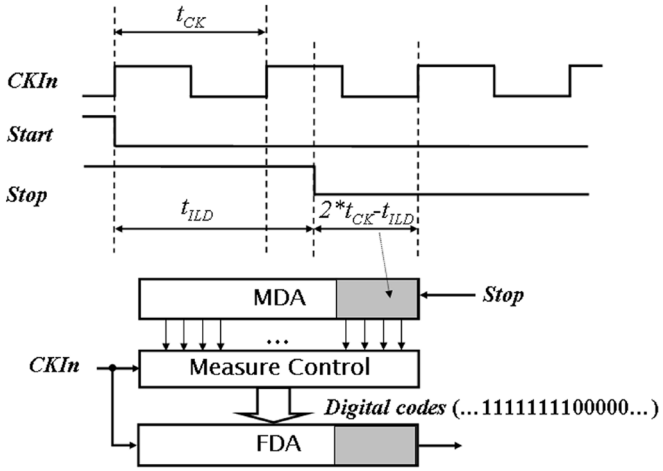


Fig. 10. Diagram for the MCD operation ($N = 2$).

in both Figs. 9 and 10, the *Start* signal propagates through the clock tree and I/O model and becomes the *Stop* signal at the input of the MDA. The time difference between the two signals (*Start* and *Stop*) is $t_{\rm ILD}$. The outputs of the MDA are sampled by the CKIn and the output digital codes are stored in the Measure Control block. After the difference between $N^*t_{\rm CK}$ and $t_{\rm ILD}$ is captured and reflected in the FDA, the initialization is completed and the loop is locked (total delay from external to output clock is $N^*t_{\rm CK}$). The output of the FDA is selected and normal DLL loop is resumed thereafter for voltage and temperature tracking. Because MDA and FDA are identical and operating mutual exclusively, the MDA and FDA can be combined to save area. A digital phase interpolation or mixing [8] out of one FDA element is followed to achieve better resolution, generally in 5–15 ps range.

Typical lock time for the entire DDLL is less than 60 cycles, depending on clock frequency, intrinsic loop delay, adjustment rate, and final resolution. Compared to the two-cycle lock of a conventional synchronous mirror delay (SMD) or clock-synchronized delay (CSD), the lock time is longer for the MCD because of switching the output via the MUX. Longer delay through the digital loop, clock path and I/O model with regarding to a shorter $t_{\rm CK}$ also requires extra time for the feedback signal to settle. Finer resolution achieved by subsequent phase interpolation further increases the lock time. The data quoted in Table I shows a lock time about 45 cycles for the DDLL, which is significantly better than the 200-cycle spec requirement. Although the finer adjustment circuitry can be designed using an analog approach, as in [5], an all-digital implementation is still adopted in the clock architecture for its simplicity, scalability, and fast lock time.

Based on the proposed clocking architecture, a multiphase generator is required to recover the edges lost due to clock division. For this design an analog DLL was selected. Diagrams for a conventional APG [3] and the proposed DCC-enabled APG are shown in Figs. 11(a) and (b), respectively. Both APGs are locked to 180-degree phase of the input reference instead of 360-degree phase for fast-locking, reduced power and area, and better linearity. One drawback for 180-degree locking is sensitivity to input duty-cycle distortion (DCD). Because the falling edge of the input signal is used to generate feedback for the phase detector (PD), any input DCD will cause a misalignment for the multiphase outputs. For the proposed MDLL-based CDN, the DCD is likely generated from the DDLL and global clock distribution network. A separated duty-cycle correction circuit is generally required in the conventional APG for high-speed operation.

Instead of using complementary phases, such as clock 180 and 270 in Fig. 11(a), the proposed APG adds an extra three delay stages to form a 9-stage VCDL. Multiple phases are tapped off from the equal length delay stages of the VCDL, as shown in Fig. 11(b). The dependency of output DCD due to input duty-cycle is removed with the proposed configuration. However, output DCD dependency still needs to be solved for the feedback signal (*FB*), which is generated by using the falling clock edges.

A fully differential dual-edge phase detector (DDPD) can be used to perform automatic DCC, as is shown in Fig. 12. Inputs *Ref* and *Ref #* are differential references, while *FB* and *FB #* are differential feedback signals. Based on transitions of

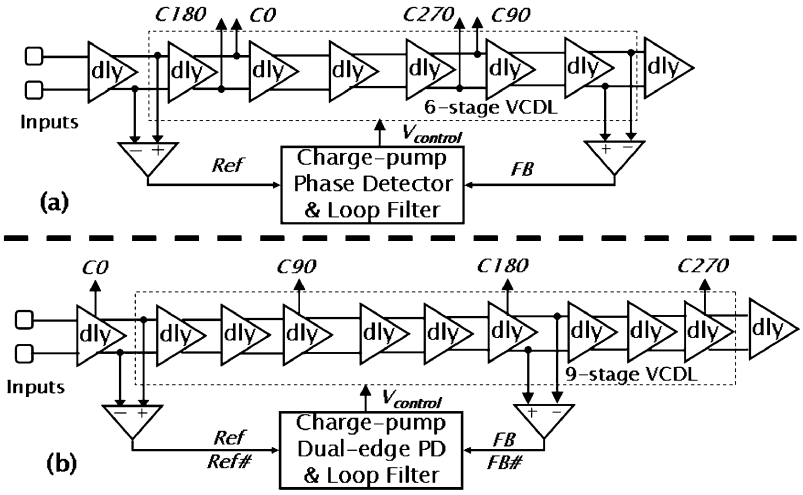


Fig. 11. (a) Conventional APG using 6-stage VCDL. (b) Proposed APG with built-in DCC.

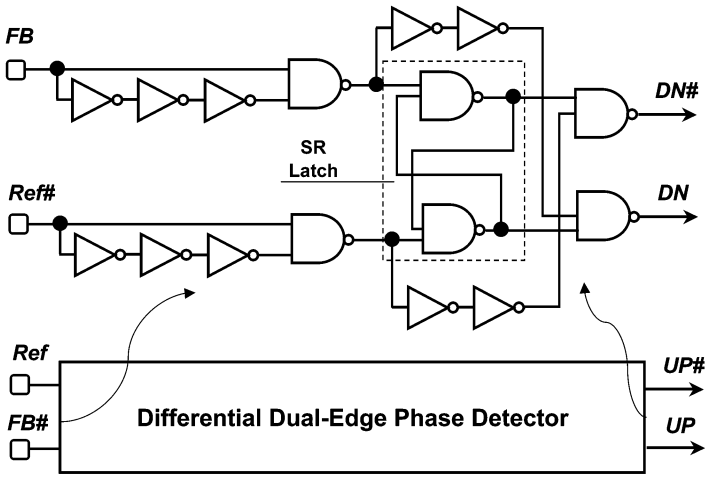


Fig. 12. Fully differential dual-edge phase detectors (DDPD).

both rising and falling edges, pulses are generated to trigger the SR latches and create *UP* and *DN* signals, respectively. Timing diagrams for this DDPD is shown in Fig. 13 with only *UP* and *DN* signals for clarity. Whenever both *UP* and *DN* are in the same state, either '1' or '0', there is no net charge dumped into the loop filter, and the control voltage $(V_{\rm control})$ remains unchanged. During normal phase locking, $V_{\rm control}$ gets adjusted twice—one at rising edges and one at falling edges—as shown in Fig. 13(a). With input duty-cycle distortion (DCD) present, the relationship is plotted in Fig. 13(b), with equal intervals for pumping up and down when locked. The average impact on $V_{\rm control}$ is zero and the loop is locked. With 1 GHz external clock and 46% duty-cycle internal clock (2 ns period), simulation data shows that the proposed APG maintains the desired phase alignment (500 ps apart from each other) while the conventional APG produced a worst-case bit time of 403 ps (40% duty cycle relative to the external clock as, measured between the four phases).

A self-bias VCDL [9] is used for the proposed APG. The VCDL is clamped to its minimum delay initially to avoid false locking. A fully differential current steering charge pump [10] is used to generate the control voltage for the VCDL with no dead zone. The typical lock time for the APG is in the order of 100 cycles. The analog implementation including all decoupling capacitors occupies roughly $133 \times 317\ \mu{\rm m}^2$, which is one-third of the size of the DDLL. Average current drawn from the APG is around 3–5 mA depending on the frequency and operating conditions.

IV. Measurement Results

Using the proposed mixed-mode DLL, an 88 $\rm mm^2$ 512 Mb $\times$ 32 GDDR3/GDDR4 device, is fabricated in a 1.5 V 3-metal 95 nm CMOS process. Fig. 14 shows the die micrograph with the digital DLL (DDLL), two analog phase generators (APG) and four quadrants of DQ paths highlighted. Fig. 15 is a measured $t_{\rm CK}$ *-VDD* shmoo data showing device performance reaching beyond 2 Gb/s/pin in GDDR3 operation for page fast write and read vectors that produce 2 ns column cycle times (data failure limited). The measured jitter histogram of read strobe (*RDQS*) is shown in Fig. 16, where the part is running at 1.5 V, 1 GHz in GDDR4 mode. The RMS jitter and the peak-to-peak jitter are 4.63 ps and 38 ps, respectively.

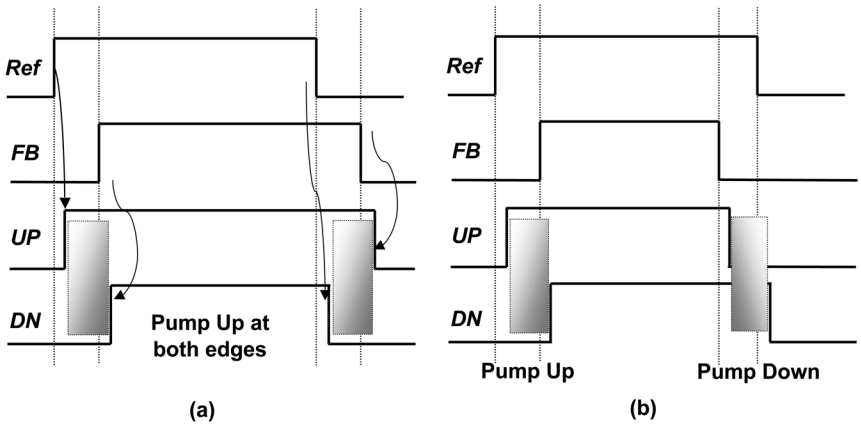


Fig. 13. Timing diagram of a dual-edge phase detector: (a) DDPD phase locking and (b) DDPD locked w/DCD.

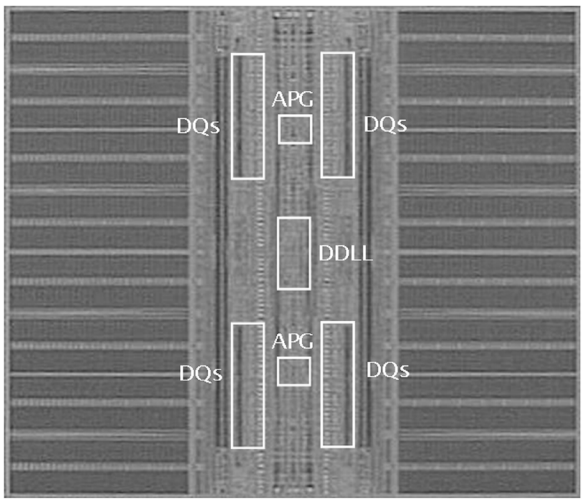


Fig. 14. Die micrograph.

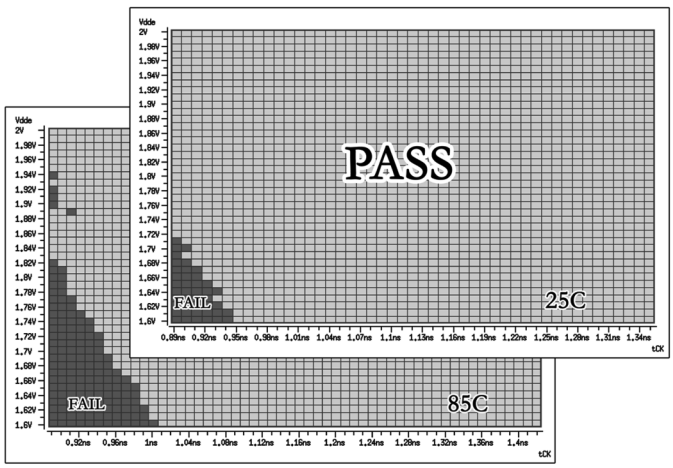


Fig. 15. t_{CK} versus *VDD* shmoo GDDR3, BL $= 4$, Temp. $= 5$ C/85 C, CL $=$ 12.

Fig. 17 is a scope shot of a DQ output running at 4 Gb/s/pin, 2.12 V supply, ambient temperature and GDDR4 mode. The data rate can be extended well beyond 2.5 Gb/s by using the proposed half-speed clocking and mixed-mode DLL. Performance results are summarized in Table II.

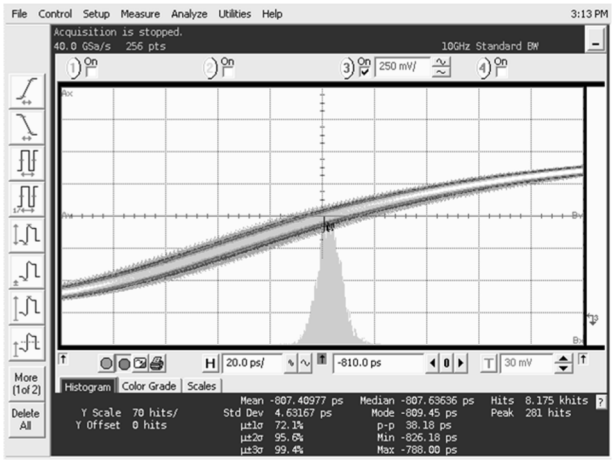


Fig. 16. Measured jitter histogram at GDDR4, 1 GHz, 1.5 V.

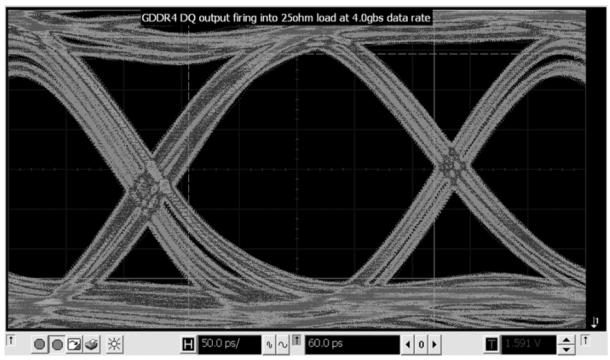
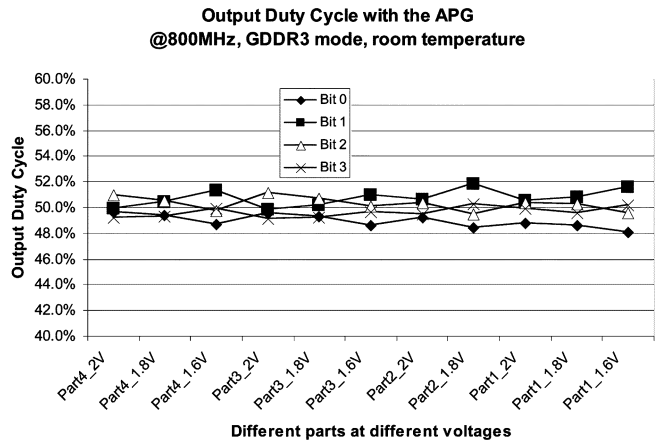


Fig. 17. GDDR4 DQ output firing into 25 ohm load at 4.0 Gb/s and 2.12 V.

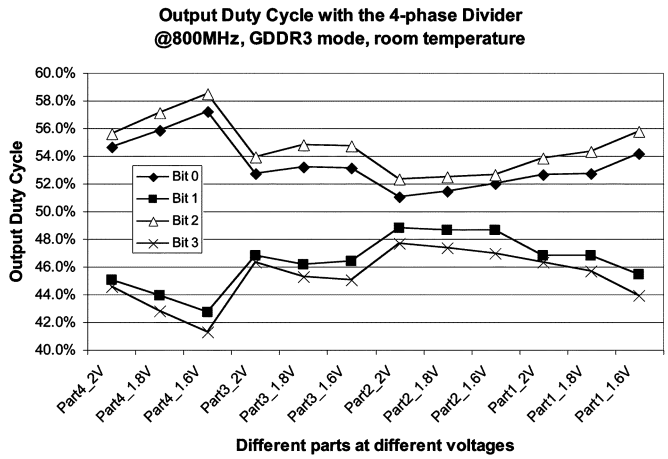
For GDDR3 mode of operation, where the full-speed clock is used internally, the proposed APG can also serve as an analog duty-cycle corrector (ADCC). Only two phases of the outputs of the APG are selected ($C0$ and $C180$ in Fig. 11(b)) and four-phase clocks are generated by clock dividers based on the rising edges of $C0$ and $C180$. Silicon results for using the ADCC or divider only are shown in Fig. 18(a) and (b), respectively.

TABLE II
PERFORMANCE SUMMARY OF THE PROPOSED MDLL

Technology	**3-metal 95nm CMOS process**
Supply Voltage	**1.5 ~ 1.8V nominal**
Operating Frequency	**400 ~ 1250 MHz**
Jitter at 1.5V, 1GHz, GDDR4	**38ps (peak-to-peak), 4.63ps (RMS)**
Lock Time	**< 200 clock cycles**
Maximum Speed	**4 Gb/s/pin at 2.12V supply**



(a)



(b)

Fig. 18(a). Silicon data for GDDR3 mode running at 800 MHz, (a) output duty cycle with the APG as a DCC, (b) output duty cycle with a 4-phase divider, no DCC.

Using ADCC improves the output duty-cycle from 41.3/58.5% to 47.3/52.1%.

V. CONCLUSION

To obtain high-speed and wide-range operating for both GDDR3 and GDDR4 SDRAM systems, a mixed-mode DLL containing both digital and analog features is proposed. The data path and command decode logic was adjusted to accommodate and take advantage of multi-phase clocks generated by the MDLL. A unique built-in DCC-enabled analog DLL reduces the design complexity by removing a dedicated DCC loop. A clock distribution network, based on the proposed MDLL, was able to achieve low-power, low duty-cycle distortion and robust operation over a wide range of clock frequency. The proposed clocking techniques can also be easily transferred to future high-speed DDR SDRAM generations.

REFERENCES

[1] F. Lin, J. Miller, A. Schoenfeld, M. Ma, and R. J. Baker, "A register-controlled symmetrical DLL for double-data-rate DRAM," *IEEE J. Solid-State Circuits*, vol. 34, no. 4, pp. 565–568, Apr. 1999.

[2] T. Matano, Y. Takai, T. Takahashi, Y. Sakito, I. Fujii, Y. Takaishi, H. Fujisawa, S. Kubouchi, S. Narui, K. Arai, M. Morino, M. Nakamura, S. Miyatake, T. Sekiguchi, and K. Koyama, "A 1-Gb/s/pin 512-Mb DDRII SDRAM using a digital DLL and a slew-rate-controlled output buffer," *IEEE J. Solid-State Circuits*, vol. 38, no. 5, pp. 762–768, May 2003.

[3] S. Sidiropoulos and M. A. Horowitz, "A semi-digital dual delay-locked loop," *IEEE J. Solid-State Circuits*, vol. 32, no. 11, pp. 1683–1692, Nov. 1997.

[4] Y. Jung, S. Lee, D. Shim, W. Kim, C. Kim, and S. Cho, "A dual-loop delay-locked loop using multiple voltage-controlled delay lines," *IEEE J. Solid-State Circuits*, vol. 36, no. 5, pp. 784–791, May 2001.

[5] J. Kim, S. Lee, T. Jung, C. Kim, S. Cho, and B. Kim, "A low-jitter mixed-mode DLL for high-speed DRAM applications," *IEEE J. Solid-State Circuits*, vol. 35, no. 10, pp. 1430–1436, Oct. 2000.

[6] G. Dehng, J. Lin, and S. Liu, "A fast-lock mixed-mode DLL using a 2-b SAR algorithm," *IEEE J. Solid-State Circuits*, vol. 35, no. 10, pp. 1464–1471, Oct. 2001.

[7] B. Johnson, B. Keeth, F. Lin, and H. Zheng, "Phase-tolerant latency control for a combination 512 Mb 2.0 gb/s/pin GDDR3 and 2.5 gb/s/pin GDDR4 SDRAM," in *IEEE ISSCC Dig. Tech. Papers*, 2007, pp. 494–495.

[8] B. Garlepp, K. S. Donnelly, K. Jun, P. S. Chau, J. L. Zerbe, C. Huang, C. V. Tran, C. L. Portmann, D. Stark, Y. F. Chan, T. H. Lee, and M. A. Horowitz, "A portable digital DLL for high-speed CMOS interface circuits," *IEEE J. Solid-State Circuits*, vol. 34, no. 5, pp. 632–642, May 1999.

[9] J. Maneatis, "Low-jitter process-independent DLL and PLL based on self-biased techniques," *IEEE J. Solid-State Circuits*, vol. 31, no. 11, pp. 1723–1732, Nov. 1996.

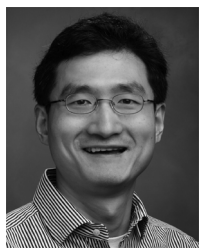
[10] K. Wong, E. Fayneh, E. Knoll, R. Law, C. Lim, R. Parker, F. Wang, and C. Zhao, "Cascaded PLL design for a 90 nm CMOS high-performance microprocessor," in *IEEE ISSCC Dig. Tech. Papers*, 2003, pp. 422–424.

[11] D. Harris, *Skew Tolerant Circuit Design*. San Francisco, CA: Morgan Kaufman, 2001.

[12] H. Partovi, "Clocked Storage Elements," in *Design of High-Performance Microprocessor Circuits*, A. Chandrakasan, W. J. Bowhill, and F. Fox, Eds. Piscataway, NJ: IEEE Press, 2001.

[13] H. J. M. Veendrick, "The behavior of flip-flops used as synchronizers and prediction of their failure rate," *IEEE J. Solid-State Circuits*, vol. 15, no. 2, pp. 169–176, Apr. 1980.

[14] F. Gray, "Pulse code communication," U.S. Patent 2,632,058, Mar. 17, 1953.



Feng (Dan) Lin (M'04) was born in Fuzhou, China, in 1971. He received the B.S. and M.S. degrees from the University of Electronic Science and Technology of China, Chengdu, China, in 1992 and 1995, respectively, and the Ph.D. degree from the University of Idaho, Boise, in 2000, all in electrical engineering.

He joined DRAM Design R&D at Micron Technology, Inc., Boise, in 2000 and currently is a Senior Design Engineer involved in the development of high-speed, leading edge DRAM for graphics and high-performance computing. He is a coauthor of the textbook *DRAM Circuit Design-Fundamental and High-Speed Topics* (Wiley-IEEE Press, 2007). He holds over 50 U.S. and foreign patents to date. His research interests include high-speed I/O circuits, PLL/DLL, and mixed-signal circuit design.

Roman A. Royer was born in Schladming, Austria, in 1979. He received the B.S. degree in electrical engineering and computer science engineering from LeTourneau University, Longview, TX, in 2001.

He joined Micron Technology Inc., Boise, ID, in 2001 as a Product Engineer. Since then, he has worked on the development and failure analysis of DDR2, DDR3, GDDR3, and GDDR4 products. He has patent applications relating to memory data path architecture and multi-phase clock generation. His interests include high-speed clock generation and distribution, latency control, on-die temperature sensing, and output drive calibration, as well as automated simulation input vector generation.

Brian Johnson (M'98) received the B.A. degree from Whitman College in 1988, and the B.S.E.E. and M.S.E.E. degrees from the University of Idaho, Boise, in 1996 and 2002, respectively.

He is a Senior Design Engineer in DRAM Design R&D at Micron Technology Inc., Boise, involved in the development of high-speed, leading edge DRAM for graphics and high-performance computing. He holds over 60 U.S. and foreign patents. He coauthored the textbook *DRAM Circuit Design-Fundamental and High-Speed Topics* (Wiley-IEEE Press, 2007).

Brent Keeth (M'82) received the the B.S.E.E. and M.S.E.E. degrees from the University of Idaho, Boise, in 1982 and 1996, respectively.

He is a Micron Fellow in DRAM Design R&D at Micron Technology Inc., Boise, responsible for the development of high-speed, leading-edge DRAM for graphics and high-performance computing. Prior to joining Micron in 1992, he held various design engineering positions at Texas Instruments, General Instruments, and Grass Valley Group. His 25 years of industry experience spans radar and avionics components, communication systems, broadcast television equipment, and solid-state memory. He holds over 300 U.S. and foreign patents to date. He coauthored the textbooks *DRAM Circuit Design—A Tutorial* (Wiley-IEEE Press, 2001) and *DRAM Circuit Design-Fundamental and High-Speed Topics* (Wiley-IEEE Press, 2007).

Mr. Keeth periodically reviews articles for publication in the IEEE JOURNAL OF SOLID-STATE CIRCUITS and has served numerous times on program committees for both the IEEE International Solid-State Circuits Conference (ISSCC) and the Symposium on VLSI Circuits. He is also a member of the University of Idaho Engineering College Advisory Board, the University of Idaho ECE Advisory Board, the Utah State University Micron Research Center Advisory Board, and the Bishop Kelly Foundation Board of Directors.