ISSN 23499842(Online), Volume 1, Special Issue 2(ICITET 15), March 2015
International Journal of Innovative Trends and Emerging Technologies

IMPLEMENTATION OF DLL TO REDUCED JITTER IN DRAM

A. LOGANATHAN[1], R. RAMESH[2]
[1]PG Student, VLSI Design, [2]Assistant Professor (Sr.G), [1,2]Dept. of ECE,
[1,2]Surya Group of Institutions, Vikiravandi, India

Abstract*:*
Modern systems use synchronous communication to achieve high data transmission rates to and from the DRAMs in the memory system. Systems that communicate synchronously use a clock signal as a timing reference so that data can be transmitted and received with a known relationship to this reference. A digitally controlled DLL is a good solution Analysis of Delay Locked Loop Used in DRAM Interfaces. It has fast wake-up time. However, digitally controlled DLLs have poor PSRR that cause large additive jitters in comparison with analog types. In order to improve the jitter performance for the operating modes and to reduce the power consumption for the power down modes, an analog DLL with a DAC, a mixed-type DLL In this paper presents the application feasibility of mixed mode PLL-DLL in DRAM. Jitter analysis of mixed mode PLL-DLL in DRAM environment has been carried out. According to the jitter type, this model can be used as pure. PLL or pure DLL or a mixed PLL-DLL. It is observed that mixed mode PLL-DLL architecture can combine the advantage from both PLL and DLL to reduce jitter in DRAM.

I. INTRODUCTION

Modern systems use synchronous communication [1] to achieve high data transmission rates to and from the DRAMs in the memory system. Systems that communicate synchronously use a clock signal as a timing reference so that data can be transmitted and received with a known relationship to this reference. A difficulty in maintaining this relationship is that process, voltage, and temperature variations can alter the timing relationship between the clock and data signals, resulting in reduced timing margins. This problem gets worse as signaling speeds increase, limiting the ability of systems to communicate data at higher speeds. A DLL is used to maintain the timing relationship between a clock signal and an output data signal. A critical element of the DLL is the phase detector, which detects phase differences between the clock and output data. The phase detector detects this phase difference, and sends control information through a low pass filter to a variable delay line that adjusts the timing of the internal clock to maintain the desired timing relationship (PLLs use a voltage controlled oscillator to adjust this timing relationship). One of the difficulties of maintaining phase

Relationships between these two signals is that the loop which provides feedback to the phase detector must account for the timing characteristics of the output logic

and output driver. This is important, as it estimates the phase differences between the clock and the data being driven by the output driver. In order to accomplish this, circuits that mimic the behavioral characteristics of the output logic and output driver are inserted into this feedback loop to model timing delays and changes in behavior as process, voltage, and temperature vary. Maintaining the timing relationships between the clock and output data in this manner with DLLs and PLLs results in improved timing margins and addresses an important limitation. Generally, an analog DLL has difficulties satisfying requirements for fast wake-up time

from various power down modes with low power consumption, and a digital-to-analog converter (DAC) was adopted for the DRAM DLL[2] to address this issue. DLL control schemes can be divided into three categories: analog, digital, and a mixed. Generally, analog schemes with current mode logic (CML) delay line have good power supply rejection ratios (PSRRs) with jitter. However, the power consumption is relatively higher compared with the digital type. A digitally controlled DLL is a good solution Analysis of Delay Locked Loop Used in DRAM Interfaces. It has fast wake-up time. However, digitally controlled DLLs have poor PSRR that cause large additive jitters in comparison with analog types. In order to improve the jitter performance for the operating modes and to reduce the power consumption for the power down modes, an analog DLL with a DAC, a mixed-type DLL[3] control scheme, is a good alternative.

According to operating specifications, DRAMs can be categorized into two classes: main memory DRAM and graphics DRAM. The former includes DDR1, DDR2, DDR3, and DDR4, and the latter includes GDDR3, GDDR4, and GDDR5[5]. The recently developed GDDR5 does not employ DLLs, and DDR4 is underdevelopment. Therefore, the DLLs in DDR1, DDR2, DDR3, GDDR3, and GDDR4 are the main focus of our analysis.

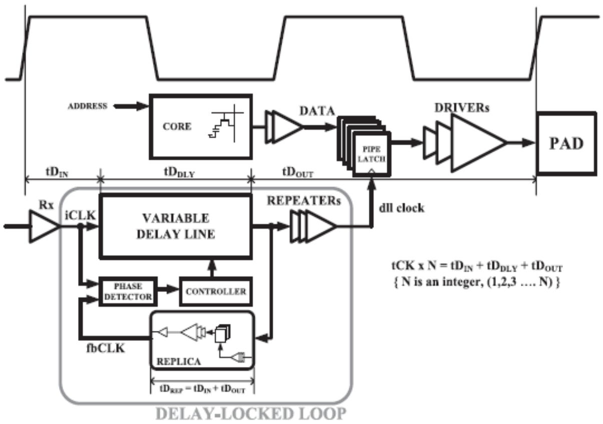


Fig 1.1 DLL structure

II. RELATED WORK

A 256-Mb synchronous DRAM with a stabilized clock access time of 1 ns was developed using an RDLL circuit. This part takes 0.1% of the entire chip area and consumes only 10 mW. The external power supply is 3.3 V, while the internal voltage is regulated to 2.5 V. Low-threshold voltage transistors used for sense amplifiers facilitated the 1.8 V low voltage margin. The DRAM is 14.91 22.01 mm using the 0.28- m design rules. The SBC cell structure enables fabrication of cells in exact 8F-square, thus making the design rules tolerant enough to implement this technology in the mass production of 0.28- to 0.24- m DRAM's.[1] a delay-locked loop (DLL) circuit having two advancements, a dual-loop operation for a wide lock range and programmable replica delays using anti-fuse circuitry and internal voltage generator for a post-package skew calibration. The dual-loop operation uses information from the initial time difference between reference clock and internal clock to select one of the differential internal loops. This increases the lock range of the DLL to the lower frequency. In addition, incorporation of the programmable replica delay using anti-fuse circuitry and the internal voltage generator allows for the elimination of skews between external clock and internal clock that occur from on-chip and off-chip variations after the package process. The proposed DLL, fabricated on 0.16- m DRAM process[2], operates over the wide range of 42–400 MHz with 2.3-V power supply. The measured results show 43-ps peak-to-peak jitter and 4.71-ps rms jitter consuming 52 mW at 400 MHz. an all-digital delay-locked loop (DLL) which achieves low jitter and stable duty cycle correction (DCC) operation. Since the DLL has dual DCC circuit, with the combinations of two DCC circuits, the DLL can correct 12.9% and 6.13% duty error under 2% at 333 MHz with 1.6 V. The DLL operates up to 1.67 GHz with 1.8 V and 1.78 GHz with 2.0 V supply voltage, and its peak-to-peak jitter at 1.4 GHz with 1.8 V is 29 ps. The power dissipations are 4.2 mW (5 mW) at 100 MHz and 19.8 mW (29.5 mW) at 1 GHz with 1.5 V (1.8 V) supply voltage with the help of the update gear circuit from the previous work. And the DLL is fabricated with 54-nm DRAM CMOS technology. The active area of the DLL is 0.11 mm[4]. A 512 M-bit consumer DDR2 SDRAM that uses self-dynamic voltage scaling (SDVS) and adaptive design techniques is introduced in this paper. With the increase in the significance of process variation, higher performance requirements reduce the allowable design margin in DRAM circuits [5]. However, self-dynamic voltage scaling gives a greater timing margin in the circuitry by changing the internal supply voltage in response to the operating frequency and process skew. By changing the internal supply voltage, the life time of the chip increases by more than 23 times when the supply voltage is lowered by 300 mV. The proposed adaptive design techniques include an adaptive bandwidth delay-locked loop and an adaptive clock gating. The former improves the performance by obtaining a wider valid data window and the latter saves on dynamic power consumption in the clock distribution network. The SDVS method reduces the IDD3P by 9.3% and the adaptive clock gating saves 8.8% of the IDD3N when measured at 200 MHz, 25 C The studied consumer DDR2 SDRAM was fabricated using 44 nm standard DRAM process technology. It occupies a 17.7 mm die area and operates using a 1.8 V power supply. DRAMs with DLL that adopt the racing mode and the countered CAS latency controller. The OA-DCC for duty cycle correction is adopted. The proposed DLL can save dynamic power consumption using the MDCDL. MDCDL shares the forward delay gates with the other delay paths. The shared delay gates are small, if the activated delay gates are small. However, the more frequently a delay path is activated, the more the delay gates are shared. Therefore, the more that dynamic power dissipates, the more dynamic power is saved. Additionally, the racing mode for DLL can reduce the dynamic power consumption by selecting the DLL with the shorter delay path between the two DLLs.[6 The dynamic power dissipation of the delay line is reduced by 68% by MDCDL and racing mode. OR-AND functioned DCC is controlled by digital logic. Therefore, it is very easy to increase the DCC capability. The proposed DLL consumes less power than that of the previous work. [7]L

III. PROPOSED DELAY LOCKED LOOP

a) *Hybrid DLL model*

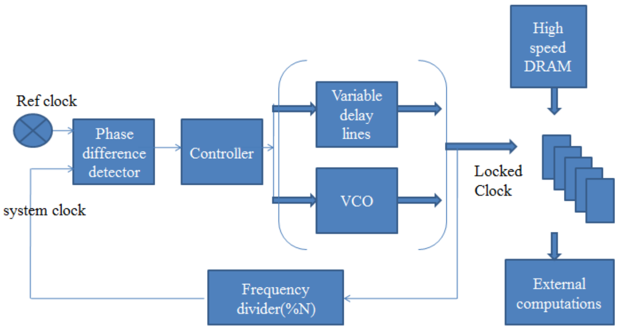


Fig 3.1 Block diagram of Hybrid DLL Model

a) *Phase detector*

A phase detector or phase comparator is a frequency mixer, analog multiplier or logic circuit that generates a voltage signal which represents the difference in phase between two signal inputs. It is an essential element of the phase-locked loop (PLL). Phase detector will detect the phase difference between system clock and phase shifted clock. Based on the phase error we use delay lines to change the phase of phase shifted clock till we get phase error as zero. Once phase matched then ring oscillator job is over.

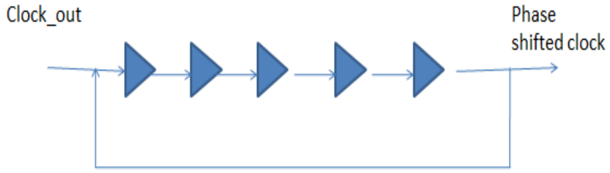


Fig 3.2 Phase detector using ring oscillator

b) Variable delay line

Here we used configurable delay lines to generate phase shift in the output clock till the output clock get matched with system clock. In delay lines N-number of delay elements are connected serially. N-should be odd. Each element has its own delay. Here in simulator tool won’t consider any delay. For simulation purpose we assign 1ps as a propagation delay for each element. Then we keep on sending the clock through this till the phase get matched with system clock. Here we use DLL which will generate various frequency range with the same phase with source clock. Clock divider is used to divide the source clock .Here reconfigurable clock divider will be used along with reconfigurable delay lines based phase match schemes. So this model works as a phase detector which can be used for phase match to the reference clock. By using this model single source clock can be used for multi rate clock domains.

c) Variable dram

i) DRAM

Dynamic random-access memory (DRAM) is a type of random-access memory that stores each bit of data in a separate capacitor within an integrated circuit. The capacitor can be either charged or discharged; these two states are taken to represent the two values of a bit, conventionally called 0 and 1. Since even non-conducting transistors always leak a small amount, the capacitors will slowly discharge, and the information eventually fades unless the capacitor charge is refreshed periodically. Because of this refresh requirement, it is a dynamic memory as opposed to SRAM and other static memory.

ii) Different DRAM

As shown in figure 3.1, Here six DRAM with various operating clocks are implemented based on single source clock by using clock divider. All phase signals will matched with DLL.

d) Steps used for locked clock

i) Source clock will be divided into required range .

ii) This divided clock will be matched with source clock using DLL.

iii) For every positive clock of match data will be read out from each DRAM.

iv) During matching time no output will be read out from any DRAM.

e) *Phase frequency detector component*

The PFD block has the task of representing the phase and frequency difference between the two input signals FR and FV/N in voltage or current form. The exact operation of the PFD differs depending on the type of PFD; it may be a mixer (multiplier) phase detector, an EXOR gate, a charge pump phase frequency detector (CP-PFD), or an edge triggered JK flip-flop [1]. The mixer phase detector is used to determine the phase difference between two analogue signals, while the rest are generally used for digital signal comparison

g) Voltage controlled oscillator component

The VCO component block uses the loop filter output to generate an output signal with a frequency of FV, this can be realised by a number of different methods, the most common of these is the LC tank oscillator which provides a much purer output signal than other VCO architectureThe ideal VCO output frequency is determined using equation(1)

$$FV = KVVC + FFR \ (1)$$

where KV is the VCO gain, and FFR is the VCO free running frequency (or the output frequency when the input control voltage VC is zero).

h) Feedback Divider Component

As mentioned earlier the feedback frequency divider (÷N) is used in frequency synthesis to produce a PLL output signal frequency FV that is some multiple, N, of the reference signal, FR. There are two common classes of feedback divider: the integer-N and fractional-N dividers. The integer-N type divides the frequency by an integer divide ratio, while the fractional-N divider provides a fractional divide ratio. The fractional divide ratio is achieved by selecting between integer values, for example, if a divide ratio of 10.5 is required, then the fractional divider could divide by 10 for 50% of the time and by 11 for 50% of the time [6]. The variation in the divide ratio can be achieved by modulating the divider through the use of random jittering, phase interpolation, or a sigma delta modulator (SDM) [7] The advantages of a finer frequency resolution with the fractional divide ratio are valuable in many modern applications.

DPLL is designed as follows:
1. The loop filter parameters are selected based on the desired system bandwidth, ωc.
2. The VCO and PFD gain components are selected using 'rule of thumb'.
3. The PLL system is simulated to check the system transients and noise band limit.
4. The loop bandwidth, ωc and the gains are reselected to vary the system transients and band limit as required, until the optimum system is found.
5. The system is simulated again (return to 3) until the desired system performance is achieved.

IV. SIMULATION RESULT

A) Phase error matching output

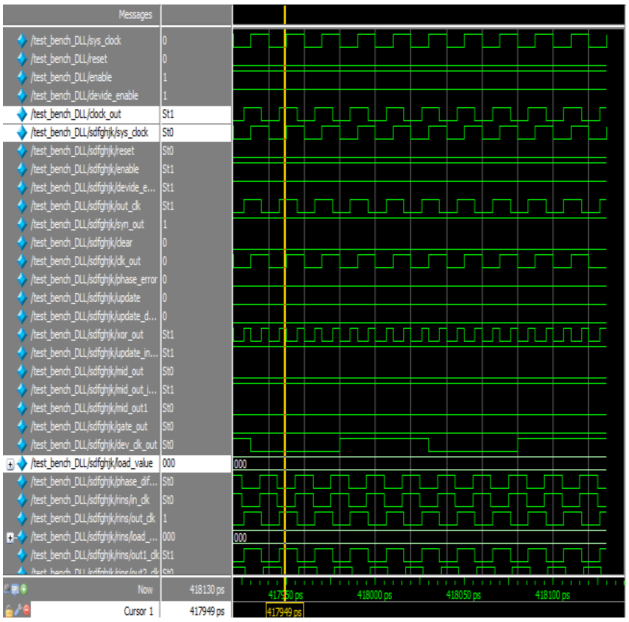

Messages
/test_bench_DLL/sys_clock 0
/test_bench_DLL/reset 0
/test_bench_DLL/enable 1
/test_bench_DLL/devide_enable 1
/test_bench_DLL/clock_out St1
/test_bench_DLL/sdfghjk/sys_clock St0
/test_bench_DLL/sdfghjk/reset St0
/test_bench_DLL/sdfghjk/enable St1
/test_bench_DLL/sdfghjk/devide_e... St1
/test_bench_DLL/sdfghjk/out_clk St1
/test_bench_DLL/sdfghjk/syn_out 1
/test_bench_DLL/sdfghjk/clear 0
/test_bench_DLL/sdfghjk/clk_out 0
/test_bench_DLL/sdfghjk/phase_error 0
/test_bench_DLL/sdfghjk/update 0
/test_bench_DLL/sdfghjk/update_d... 0
/test_bench_DLL/sdfghjk/xor_out St1
/test_bench_DLL/sdfghjk/update_in... St1
/test_bench_DLL/sdfghjk/mid_out St0
/test_bench_DLL/sdfghjk/mid_out_i... St1
/test_bench_DLL/sdfghjk/mid_out1 St0
/test_bench_DLL/sdfghjk/gate_out St0
/test_bench_DLL/sdfghjk/dev_clk_out St0
/test_bench_DLL/sdfghjk/load_value 000
/test_bench_DLL/sdfghjk/phase_dif... St0
/test_bench_DLL/sdfghjk/ins/in_clk St0
/test_bench_DLL/sdfghjk/ins/out_clk 1
/test_bench_DLL/sdfghjk/ins/load_... 000
/test_bench_DLL/sdfghjk/ins/out1_clk St1
000
000
Now 418130 ps
Cursor 1 417949 ps
417950 ps
418000 ps
418050 ps
418100 ps
417949 ps

Fig 4.1 For load_value=0 phase error in XOR_OUT

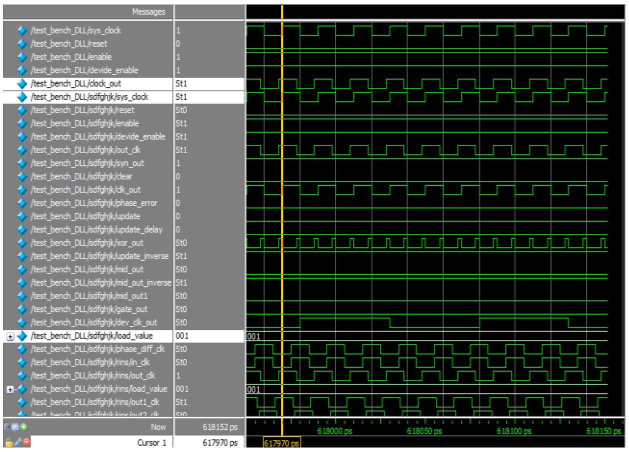

Messages
/test_bench_DLL/sys_clock 1
/test_bench_DLL/reset 0
/test_bench_DLL/enable 1
/test_bench_DLL/devide_enable 1
/test_bench_DLL/clock_out St1
/test_bench_DLL/sdfghjk/sys_clock St1
/test_bench_DLL/sdfghjk/reset St0
/test_bench_DLL/sdfghjk/enable St1
/test_bench_DLL/sdfghjk/devide_enable St1
/test_bench_DLL/sdfghjk/out_clk St1
/test_bench_DLL/sdfghjk/syn_out 1
/test_bench_DLL/sdfghjk/clear 0
/test_bench_DLL/sdfghjk/clk_out 1
/test_bench_DLL/sdfghjk/phase_error 0
/test_bench_DLL/sdfghjk/update 0
/test_bench_DLL/sdfghjk/update_delay 0
/test_bench_DLL/sdfghjk/xor_out St0
/test_bench_DLL/sdfghjk/update_inverse St1
/test_bench_DLL/sdfghjk/mid_out St0
/test_bench_DLL/sdfghjk/mid_out_inverse St1
/test_bench_DLL/sdfghjk/mid_out1 St0
/test_bench_DLL/sdfghjk/gate_out St0
/test_bench_DLL/sdfghjk/dev_clk_out St0
/test_bench_DLL/sdfghjk/load_value 001
/test_bench_DLL/sdfghjk/phase_diff_clk St0
/test_bench_DLL/sdfghjk/ins/in_clk St0
/test_bench_DLL/sdfghjk/ins/out_clk 1
/test_bench_DLL/sdfghjk/ins/load_value 001
/test_bench_DLL/sdfghjk/ins/out1_clk St1
001
001
Now 618152 ps
Cursor 1 617970 ps
618000 ps
618050 ps
618100 ps
618150 ps
617970 ps

Fig 4.2 For load_value=1: phase error in XOR_OUT

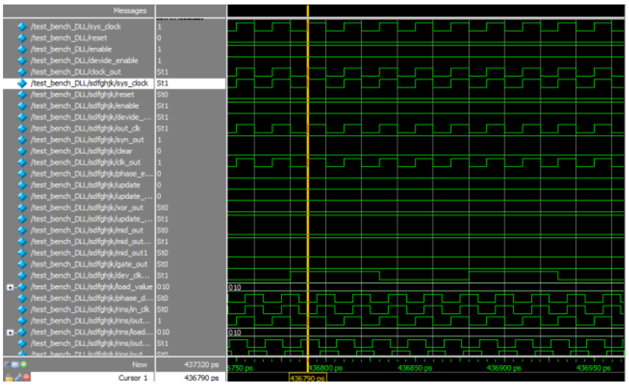

Messages
/test_bench_DLL/sys_clock 1
/test_bench_DLL/reset 0
/test_bench_DLL/enable 1
/test_bench_DLL/devide_enable 1
/test_bench_DLL/clock_out St1
/test_bench_DLL/sdfghjk/sys_clock St1
/test_bench_DLL/sdfghjk/reset St0
/test_bench_DLL/sdfghjk/enable St1
/test_bench_DLL/sdfghjk/devide_... St1
/test_bench_DLL/sdfghjk/out_clk St1
/test_bench_DLL/sdfghjk/syn_out 1
/test_bench_DLL/sdfghjk/clear 0
/test_bench_DLL/sdfghjk/clk_out 1
/test_bench_DLL/sdfghjk/phase_e... 0
/test_bench_DLL/sdfghjk/update 0
/test_bench_DLL/sdfghjk/update_... 0
/test_bench_DLL/sdfghjk/xor_out St0
/test_bench_DLL/sdfghjk/update_... St1
/test_bench_DLL/sdfghjk/mid_out St0
/test_bench_DLL/sdfghjk/mid_out... St1
/test_bench_DLL/sdfghjk/mid_out1 St0
/test_bench_DLL/sdfghjk/gate_out St0
/test_bench_DLL/sdfghjk/dev_clk... St1
/test_bench_DLL/sdfghjk/load_value 010
/test_bench_DLL/sdfghjk/phase_d... St0
/test_bench_DLL/sdfghjk/ins/in_clk St0
/test_bench_DLL/sdfghjk/ins/out... 1
/test_bench_DLL/sdfghjk/ins/load... 010
/test_bench_DLL/sdfghjk/ins/out... St1
010
010
Now 437320 ps
Cursor 1 436790 ps
436800 ps
436850 ps
436900 ps
436950 ps
436790 ps

Fig 4.3 For load_value=2, phase error in XOR_OUT= 0 phase locked

B) output with lock cycles

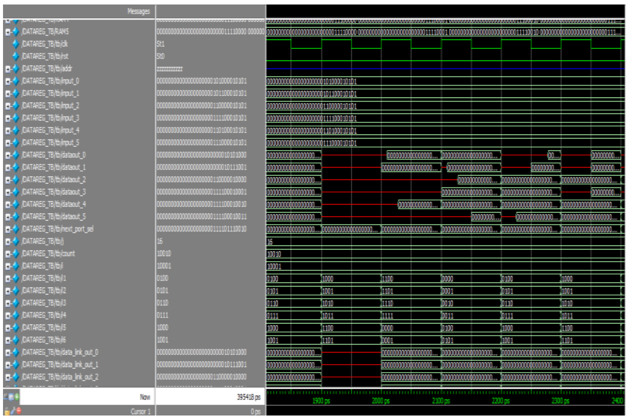
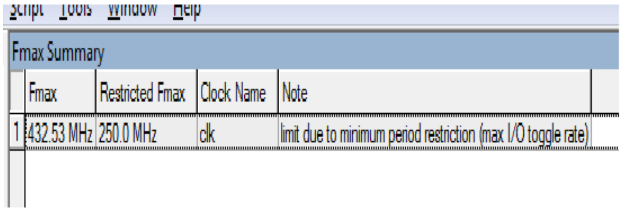


Fig 4.4 Output with delay and lock cycles

c) Frequency analysis lock cycle for maximum frequency

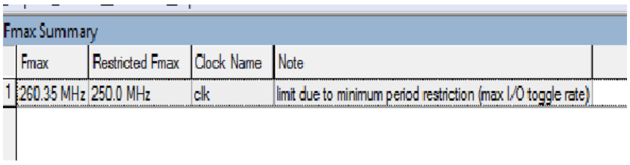


Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Note
1	432.53 MHz	250.0 MHz	clk	limit due to minimum period restriction (max I/O toggle rate)

Fig 4.5 Lock Cycle for Maximum Frequency

d) Clock cycle for minimum frequency

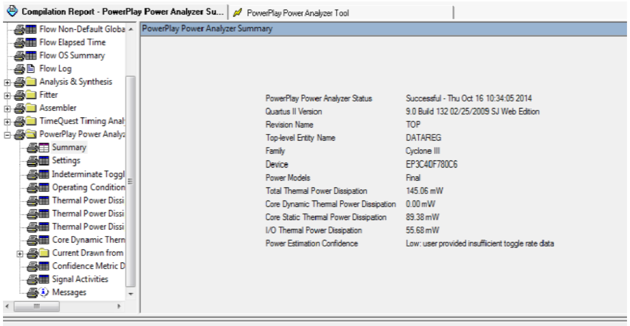


Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Note
1	260.35 MHz	250.0 MHz	clk	limit due to minimum period restriction (max I/O toggle rate)

Fig 4.5 Lock Cycle For Minimum Frequency

e) Power analysis



PowerPlay Power Analyzer Summary

PowerPlay Power Analyzer Status	Successful - Thu Oct 16 10:34:05 2014
Quartus II Version	9.0 Build 132 02/25/2009 SJ Web Edition
Revision Name	TOP
Top-level Entity Name	DATAREG
Family	Cyclone III
Device	EP3C40F780C6
Power Models	Final
Total Thermal Power Dissipation	145.06 mW
Core Dynamic Thermal Power Dissipation	0.00 mW
Core Static Thermal Power Dissipation	89.38 mW
I/O Thermal Power Dissipation	55.68 mW
Power Estimation Confidence	Low: user provided insufficient toggle rate data

Fig 4.6 Power Estimation Output

V.CONCULSION

The DLL technologies for DRAM can be analyzed in the literature between early 2000 and 2011. We described the design considerations regarding DRAM DLLs, such as lock cycles, lock range frequency, power exit time. From the analysis, the digitally controlled DLL was a more promising technology as the operating frequency increases and the power consumption requirements become more stringent.

References

[1] A. Hatakeyama, H. Mochizuki, T. Aikawa, M. Takita, Y. Ishii, H. Tsuboi, S.-Y. Fujioka, S. Yamaguchi, M. Koga, Y. Serizawa, K. Nishimura, K. Kawabata, Y. Okajima, M. Kawano, H. Kojima, K. Mizutani, T. Anezaki, M. Hasegawa, and M. Taguchi, “A 256-Mb SDRAM using a register-controlled digital DLL,” *IEEE J. Solid-State Circuits*, vol. 32, no. 11, pp. 1728–1734, Nov. 1997.

[2] S. J. Kim, S. H. Hong, J.-K. K. Wee, J.-H. H. Cho, P. S. Lee, J.-H. H. Ahn, and J.-Y. Y. Chung, “A low-jitter wide-range skew-calibrated dual-loop DLL using antifuse circuitry for high-speed DRAM,” *IEEE J. Solid-State Circuits*, vol. 37, no. 6, pp. 726–734, Jun. 2002.

[3] W.-J. Yun, H.-W. Lee, D. Shin, and S. Kim, “A 3.57 Gb/s/pin low jitter all-digital DLL with dual DCC circuit for GDDR3 DRAM in 54-nm CMOS technology,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 19, no. 9, pp. 1718–1722, Nov. 2011

[4] H.-W. Lee, K.-H. Kim, Y.-K. Choi, J.-H. Sohn, N.-K. Park, K.-W. Kim, C. Kim, Y.-J. Choi, and B.-T. Chung, “A 1.6 V 1.4 Gb/s/pin consumer DRAM with self-dynamic voltage-scaling technique in 44 nm CMOS technology,” *IEEE J. Solid-State Circuits*, vol. 47, no. 1, pp. 131–140, Jan. 2012

[5] H.-W. Lee, H. Choi, B.-J. Shin, K.-H. Kim, K.-W. Kim, J. Kim, K.-H. Kim, J.-H. Jung, J.-H. Kim, E.-Y. Park, J.-S. Kim, J.-H. Kim, J.-H. Cho, N. Rye, J.-H. Chun, Y. Kim, C. Kim, Y.-J. Choi, and B.-T. Chung, “A 1.0-ns/1.0-V delay-locked loop with racing mode and countered CAS latency controller for DRAM interfaces,” *IEEE J.Solid-State Circuits*, vol. 47, no. 6, pp. 1436–1447, Jun. 2012

[6] J. J. Kim, S.-B. Lee, T.-S. Jung, C.-H. Kim, S.-I. 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.

[7] S. Kuge, T. Kato, K. Furutani, S. Kikuda, K. Mitsui, T. Hamamoto, J. Setogawa, K. Hamade, Y. Komiya, S. Kawasaki, T. Kono, T. Amano,T. Kubo, M. Haraguchi, Z. Kawaguchi, Y. Nakaoka, M. Akiyama, Y. Konishi, and H. Ozaki, “A 0.18 μm 256-Mb DDR-SDRAM with low-cost post-mold tuning method for DLL replica,” *IEEE J. Solid-State Circuits*, vol. 35, no. 11, pp. 726–734, Nov. 2000.