All-Digital DLL Architecture and Applications

Tuvia Liran[1] and Ran Ginosar[2]
[1]DFM, Ltd., Kiryat Tivon, Israel (tuvia@dfm4vlsi.com)
[2]VLSI Systems Research Center, Technion—Israel Inst. of Technology, Haifa 32000, Israel

Abstract

An improved architecture for all digital Delay Locked Loop (ADDLL) had been developed and implemented for several applications and design methodologies. In most cases it can be based on standard cells only. Several techniques are used to minimize the jitter, achieving less than 40pS (peak) for 0.13µ technology. The frequency range is very wide, exceeding 500MHz. For 0.13µ core, the area is 0.01mm^2, and power is 2mW at 500MHz with no standby power. Scaling to smaller geometries reduces jitter, power and area. The core was fabricated and tested on several technologies. The ADDLL enables several applications, such as clock de-skewing, frequency multiplications, splitting the cycle into several phases, controlling slave delay blocks and balancing clock skew in multiple voltage domain chips.

1. Introduction

The common approach for clock processing, such as de-skewing and frequency multiplication, is based on analog Phase-Locked-Loops (PLLs) or DLLs. In recent years, the interest in all-digital DLLs (ADDLL) increased, and several papers reporting such implementations have been published [1-7].

Originally, analog implementations provided lower jitter. However, analog circuits are more difficult to scale, while digital circuits are easily scalable. In most digital implementations jitter is proportional to gate delay, and thus jitter scales down in advanced technologies. As a result, jitter in digital implementations is catching up and even surpasses that of analog implementations. A fairly low jitter, in the order of few tens of picoseconds, was reported in [8].

Analog circuits consume DC current, even at standby. In ADDLL it is possible to stop any toggling, thus no dynamic power is consumed. Since the delay setting is stored in digital storage elements, is always tuned to the required delay, and the return from standby is immediate, enabling additional power saving.

The area of ADDLL is typically much lower than that of analog DLLs. The main reason is the large filtering capacitors used in analog DLLs/PLLs. Typical ADDLLs require few thousands gates, and their area is scaled at each technology generation. The area of 0.01mm^2, reported in this paper, is much smaller than reported on analog implementations of DLLs.

Analog circuits are limited in their minimum operating voltage, since some circuits requires minimum biasing voltage for proper operation. In low voltage CMOS chips, the analog DLL may require supply voltage higher than the rest of the chip. In ADDLL, where CMOS gates are used, voltage can be scaled to very low levels, which is also the level of the other CMOS circuits.

When the delay is controlled digitally, one "master" controller can force a specific delay onto "slave" delay blocks. This feature is too sensitive in analog implementations, since the analog signals might be contaminated by noise and by supply voltage variations. This feature enables many applications, as will be described later.

Digital implementations are also much easier to port to other technologies. They are also portable to structured arrays and gate arrays, merely by routing the existing gates in the arrays.

One of the applications where ADDLL architecture is useful is in balancing clock networks in multiple-voltage domain (MVD) chips for optimizing speed and power. Tiny ADDLLs can adjust the clock dynamically as voltage is changed.

We present an improved all-digital architecture for ADDLL. It is miniature, scalable, low power, supporting many applications and easily portable to any technology. It has already been implemented as a general purpose DLL for structured ASICs, a high speed DLL for DDR interfaces, and with slave delay block and clock multiplier for custom applications.

Section 2 describes the architecture. Several applications are presented in Section 3, and performance is discussed in Section 4.

2. ADDLL architecture

2.1 Structure

A block diagram of the ADDLL is presented in Fig. 1.

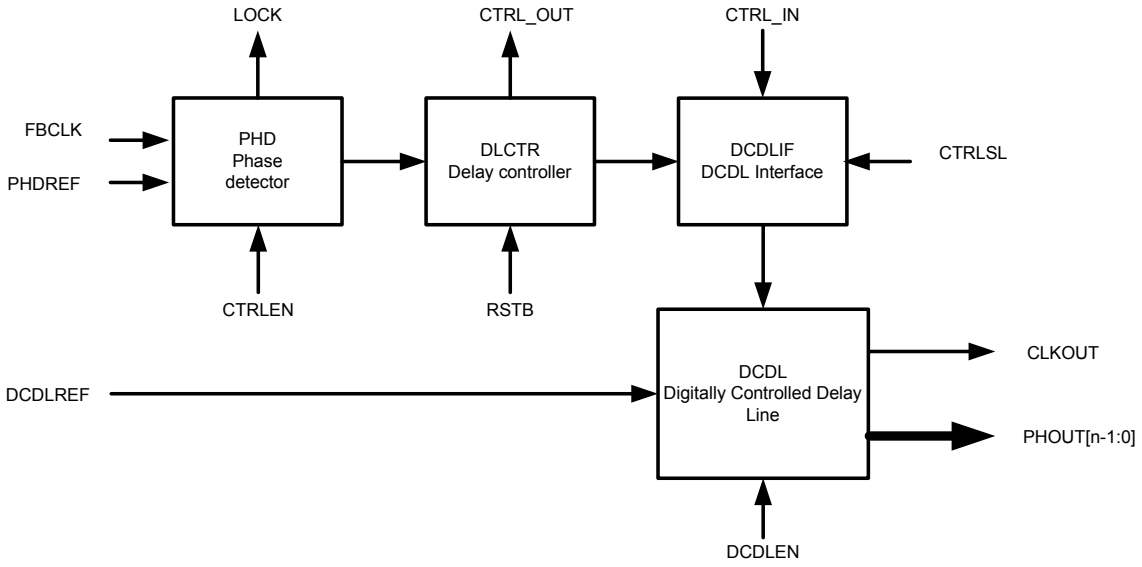

Fig. 1: ADDLL architecture

The Digitally Controlled Delay Line (DCDL) is a digital circuit whose delay is controlled by a digital control word. It consists of N identical sub-circuits, each providing 1/N of the total delay.

The Phase Detector (PHD) compares the phase between its two inputs, and generates the *UP/DOWN* signals for phase adjustment. When connecting *PHDREF* to *DCDLREF* and *CLKOUT* to *FBCLK*, the DLL will lock when the delay of DCDL will be full cycle of *DCDLREF*. Alternatively, when *FBCLK* is connected to the output of a clock tree, it will align the total skew of DCDL and the clock tree to a full *DCDLREF* cycle.

The Delay Controller (DLCTR) updates the control register, based on the *UP/DOWN* signals received from the PHD. For observability and controlling slave DCDLs, the control word is produced as an output of the block.

The DCDL Interface block (DCDLIF) selects either the internal control word (in normal operation) or an external control word.

2.2 ADDLL operation

At start-up, the control word is reset to zero, forcing minimum delay in the DCDL. The PHD detects that *FBCLK* rises before *PHDREF*, causing the DLCTR to increase the delay until it is locked. The *LOCK* signal indicates that the phase difference is small, and the DLL is locked. If the *PHDREF* is stopped, or *CTRLEN* is de-activated, the control word is not changed, and power consumption is minimized.

Each one of the N sub-circuits of the DCDL has coarse and fine delay adjustments. The coarse delay is implemented as a mirror delay line (Fig. 2). It enables the control of the delay in steps of two gate delay. The length of the delay chain is virtually unlimited, enabling operation at a very wide range of frequencies and delays. The fine delay can be implemented using any of the circuits presented in Fig. 3, depending on the cells that are available in the cell library. The fine delay enables reducing the granularity to 1/4 of a gate delay, which is about 15pS in 0.13µ technology. This fine delay scales with the technology.

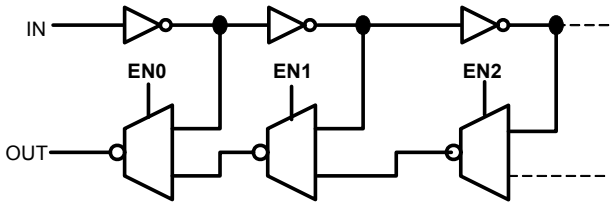

Fig. 2: Coarse delay of the DCDL

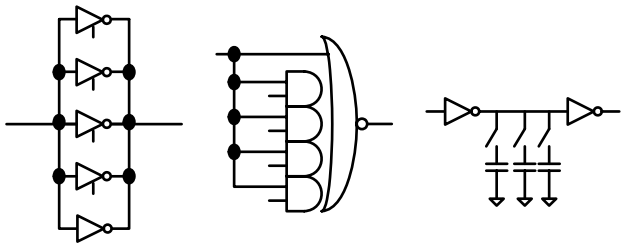

Fig. 3: Alternative fine delay lines in DCDL

An example of the DCDL response to its control word is shown in Fig. 4 for seven PVT corners, demonstrating linear dependence of delay on the control word.

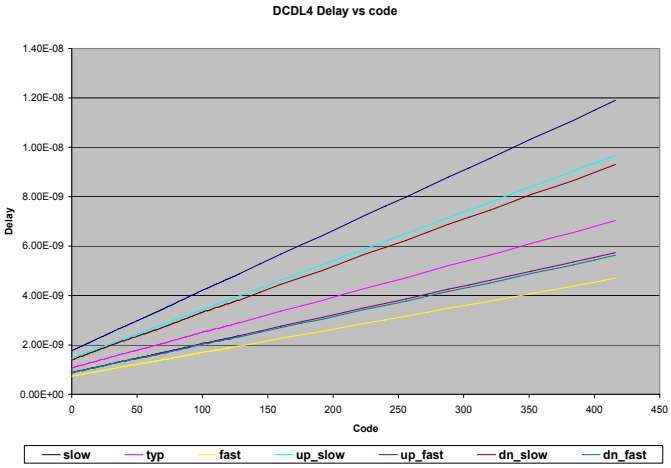

Fig. 4: DCDL response to the control word

3. Applications of ADDLL

DLLs have many applications. For some of these applications the use of ADDLL provides a significant advantage over the analog alternative. Here are few examples:

3.1 Clock de-skewing

Fig. 5 presents a typical example of using ADDLL for compensating for the skew of the clock tree.

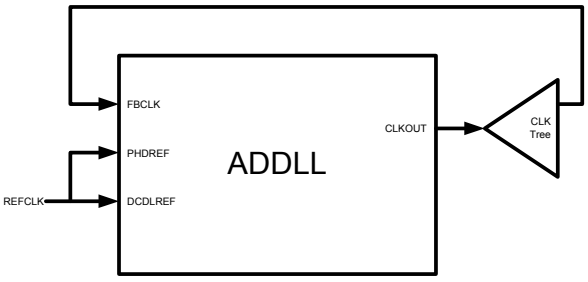

Fig. 5: Using ADDLL for clock de-skewing

3.2 Frequency multiplication

When *FBCLK* is connected to *CLKOUT*, the DCDL is locked to a full clock cycle, and each phase provides $360^{o}/N$ of the cycle. By processing the output phases of the DCDL, it is possible to generate clock with 2X or 4X of the original frequency, as presented in Fig. 6.

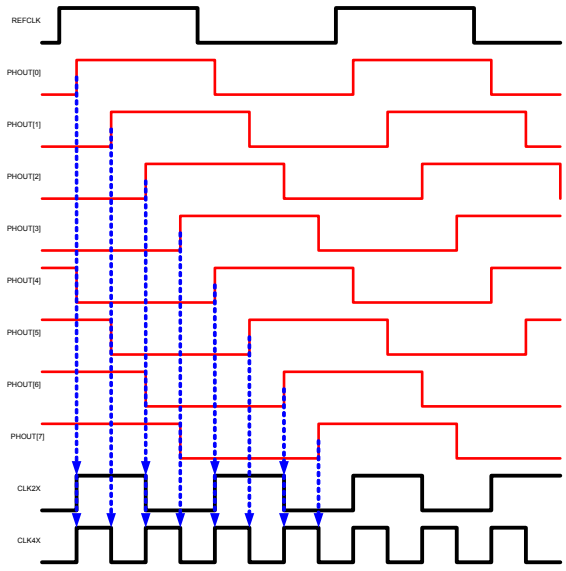

Fig. 6: Frequency multiplication by processing the output phases

3.3 Master-slave connection

By cascading several ADDLL cores, it is possible to lower the operating frequency, increase the number of phases or multiply the frequency by higher factors. An example is presented in Fig. 7.

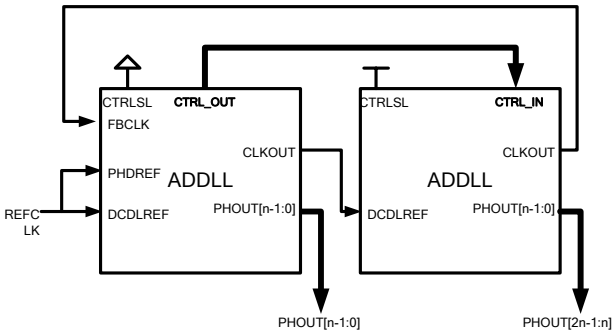

Fig. 7: Master-slave connection of two ADDLL cores

3.4 Using ADDLL for DDR2/QDR2 interfacing

In DDR2/QDR2 interfaces, there is a need to delay the data by 1/4 of clock cycle, in order to strobe the data at the center of its eye opening. This can be implemented by generating a slave delay block with 1/4 of the master's delay as presented in Fig. 8.

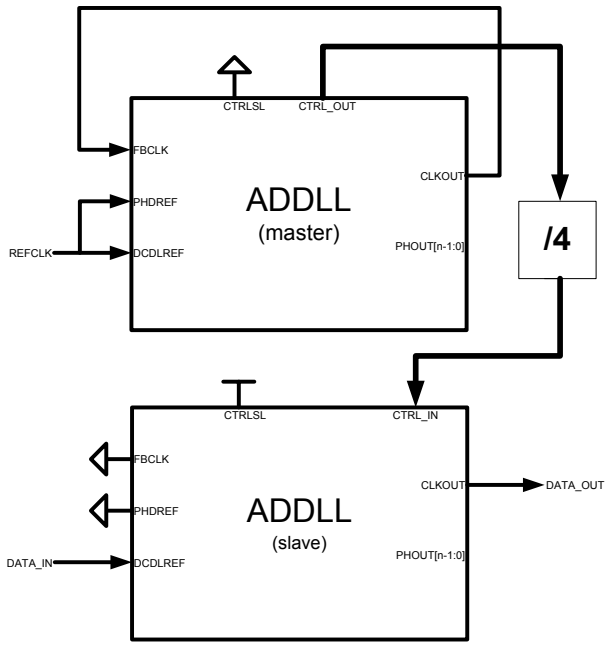

Fig. 8: Shifting a slave DCDL by 1/4 cycle for DDR/QDR applications

3.5 Clock tree balancing in multi-voltage domains

One common technique to optimize speed and power in highly integrated chips is the use multiple voltage domains (MVD) on chip. However, MVD complicates the balancing of the clock. By using a simplified version of the ADDLL in each domain it is possible to balance all clock trees dynamically, even though the voltage is different in each domain.

Fig. 9 presents the use of the simplified ADDLL in each voltage domain. *PH[0]* and *PH[1]* are two phases of the clock, generated by a global ADDLL, where *PH[1]* lags *PH[0]* by 1/4 or 1/2 cycle. These signals are distributed across the entire chip, synchronizing all clocks to *PH[1]*. When several blocks are connected to the same clock phases, as in Fig. 10, all the clock signals at the end of all branches of the clock trees are aligned with the edge of *PH[1],* even if the supply voltage is different or there is any other source of mis-match of the delay.

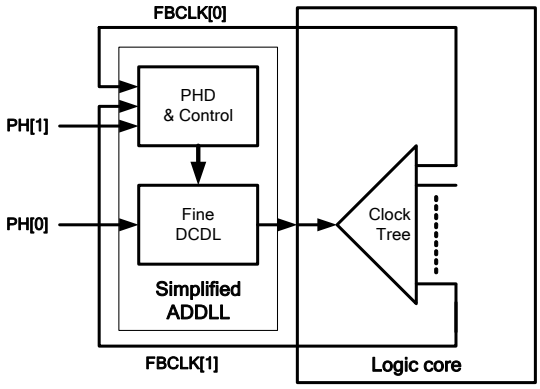

Fig. 9: Connecting a simplified ADDLL to a logic core

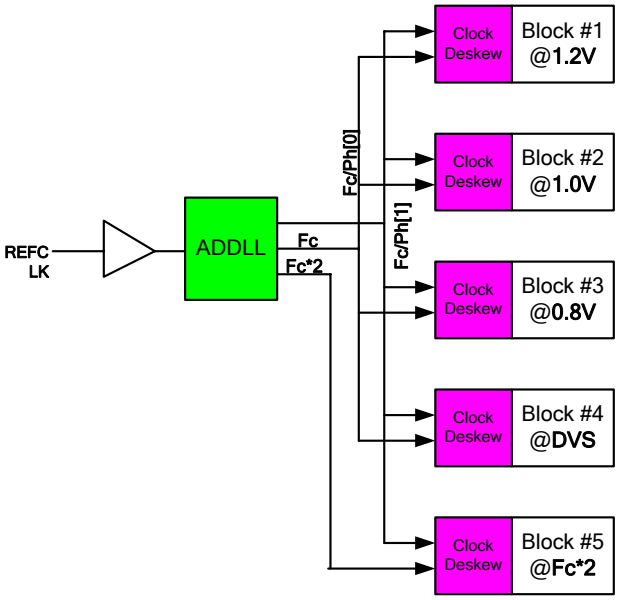

Fig. 10: Global connection of clock network for balancing clock trees in multi-voltage domains

4. ADDLL performance examples

We have successfully implemented three different versions of the ADDLL, as detailed in Table 1. One layout is demonstrated in Fig. 11.

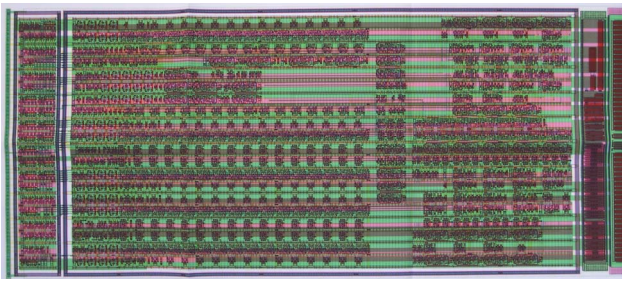

Fig. 11: layout of 0.13µ ADDLL placed in I/O ring

5. Summary

An improved architecture for ADDLL had been presented, with some of its potential applications.

The ADDLL offers an attractive solution for many clocking applications, such as de-skewing, splitting into phases, frequency multiplication, and balancing the clock skew in multi-voltage domains. Some of these applications require the use of ADDLL.

References

1. B.W. Garlepp et al.; "A Portable Digital DLL for High-Speed CMOS Interface Circuits," JSSC, 34(5):632-644, May 1999.
2. T. Olsson et al.; "A Digitally Controlled Low-Power Clock Multiplier for Globally Asynchronous Locally Synchronous Designs," ISCAS, Geneva, May 2000.
3. I.C. Hwang et al.; "A digitally Controlled Phase-Locked Loop With a Digital Phase-Frequency Detector for Fast Acquisition," JSSC, 36(10):1574-1581, Oct. 2001.
4. T. Matano et al.; "A 1-Gb/s/pin 512-Mb DDRII SDRAM Using a Digital DLL and a Slew-Rate-Controlled Output Buffer," JSSC, 38(5):762-768, May 2003.
5. M. Combes et al.; "A Portable Clock Multiplier Generator Using Digital CMOS Standard Cells," JSSC 31(7): 958-965, July 1996.
6. A. Efendovich et al.; "Multi-frequency zero-jitter delay-locked loop," JSSC 29(1):67-70, Jan 1994.
7. B.S. Kim et al.; "100MHz all-digital delay-locked loop for low power applications," Electronic letters 34(18):1739-1740, 3 Sep. 1998.
8. J. Christiansen; "An Integrated CMOS 0.15nS Digital Timing Generator for TDC's and Clock Distribution Systems,." IEEE Trans. Nuclear Science, 42(4): 753–757, Aug. 1995.

Table I: Performance of ADDLL Implementations

	I	*II*	*III*
Architecture	**ADDLL**	**Master ADDLL with a slave**	**ADDLL**
Application	**General purposes DLL**	**DCDL for phase shifting by 1/4 cycle for DDR2/QDR2**	**Clock multiplier**
Library type	**Structured ASIC**	**General purpose standard cells**	**Rad-hard standard cells**
Process	**0.18µ**	**0.13µ**	**0.18µ**
Voltage	**1.8V**	**1.2V**	**1.8V**
Frequency Range	**100-300 MHz**	**200-500 MHz**	**75-200 MHz**
Max Jitter	**60 pS**	**30 pS**	**80 pS**
Area	**0.04mm^2**	**0.014mm^2 (including slave DCDL)**	**0.05mm^2**
Power	**12mW**	**2.5mW**	**8mW**
Location on chip	**Anywhere in array**	**Within the I/O pad ring (see layout in Fig. 11)**	**Anywhere on core**