A Novel All-Digital PLL With Software Adaptive Filter

Article *in* IEEE Journal of Solid-State Circuits · April 2004

DOI: 10.1109/JSSC.2003.822780 · Source: IEEE Xplore

Liming Xiu
Institute of Electrical and Electronics Engineers
55 PUBLICATIONS 506 CITATIONS
SEE PROFILE

Jan Meiners
Unfallkrankenhaus Hamburg
21 PUBLICATIONS 203 CITATIONS
SEE PROFILE

A Novel All-Digital PLL With Software Adaptive Filter

Liming Xiu, *Senior Member, IEEE*, Wen Li, *Member, IEEE*, Jason Meiners, *Member, IEEE*, and Rajitha Padakanti, *Member, IEEE*

***Abstract*—The phase-locked loop (PLL) is one of the key building blocks of modern electronic designs. This paper presents a novel PLL structure that utilizes a "flying-adder" frequency synthesizer as its digital control oscillator (DCO), a software implemented adaptive IIR filter as its loop filter, and a unique counter as its phase detector. This all-digital PLL (ADPLL) achieved the desired functionality with additional advantages including no off-chip *R* and *C* components required, dynamic control of the loop gain on the fly, easy implementation on the digital CMOS process. This paper presents detailed descriptions of each component of this ADPLL; it also presents the system modeling in Z-domain, by mapping from S-domain, for dynamic response, stability, and steady-state error study.**

***Index Terms*—Adaptive filter, flying-adder, frequency synthesizer, phase detector, phase-locked loop.**

I. Introduction

Most present digital electronic systems are designed in synchronous design style. For mixed-signal designs with analog components (ADC, DAC, etc.) on chip, the analog components usually also work synchronously with clock edge. Clock generation is one of the most important issues in modern VLSI designs and the phase-locked loop (PLL) is the key element in this puzzle. For example, Fig. 1 is the block diagram of a graphic digitizer system that converts analog video waveforms to digital signals for LCD monitors and digital TV applications. Both the digital blocks and the ADCs operate on various clocks generated by the on-chip PLL.

In this system, the PLL needs to lock to the horizontal synchronization (HS) signal and generate the pixel clock for the ADCs, and other clocks for various digital function blocks. This PLL can be constructed as a conventional PLL with an analog voltage-controlled oscillator (VCO) and an analog loop filter. Alternatively, it can also be designed in pure digital domain with noticeable benefit in terms of cost, design effort, and flexibility. This paper will demonstrate an all-digital PLL (ADPLL) that has certain advantages over conventional PLLs for this application. The term "all-digital PLL" is used for a particular reason: ***all*** signals within this PLL are digital values; no analog voltage level is used anywhere in the system.

In this paper, Section II gives the description of each functional block in this system. Section III gives the modeling of this PLL system in both S-domain and Z-domain. Section IV is the steady-state error study. The real ADPLL implementation is shown in Section V.

Manuscript received May 6, 2003; revised November 17, 2003.

The authors are with Texas Instruments Inc., Dallas, TX 75265-5303 USA (e-mail: limingxiu@ti.com).

Digital Object Identifier 10.1109/JSSC.2003.822780

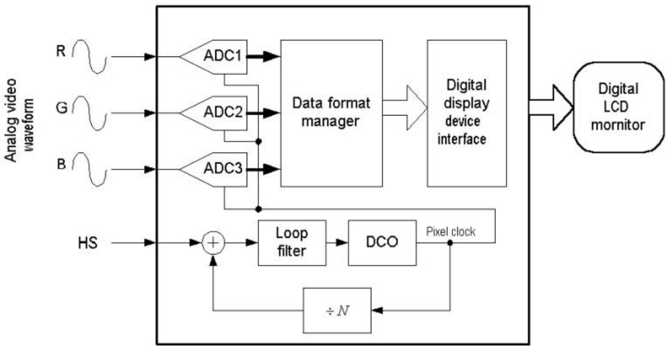


Fig. 1. Graphic digitizer system.

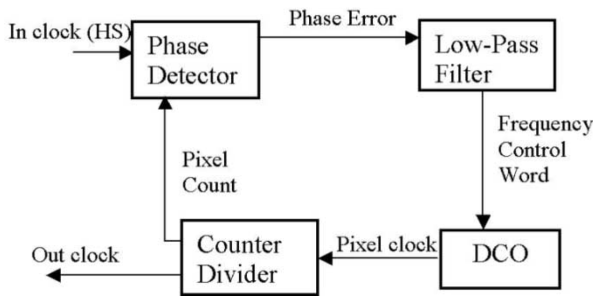


Fig. 2. Block diagram of the ADPLL.

II. Components of The ADPLL

A. Second-Order System

Fig. 2 is the block diagram of this ADPLL. As shown, it is a second-order feedback system due to the first-order low-pass filter. In this system, our goal is to generate a pixel clock whose frequency is the frequency of the HS multiplied by the pixel number of one video line. The operation of this PLL is based on sampled data, so it should be treated in Z-domain. The phase error and frequency control word are updated at the rate of HS. The digital control oscillator (DCO) is a frequency synthesizer that takes a digital value as input and generates a corresponding frequency. Each functional block will be described in the following sections.

B. Phase Detector

1) Principle Idea: In the ADPLL system of Fig. 2, the phase detector is used to measure the time difference between the external time base (HS) and the internal time base (pixel clock).

This time difference is expressed as phase error that will be fed downstream to the filter. The counter/divider is a binary counter of certain size whose upper limit can be denoted as max_pix, which is usually set to the value of ***number of pixels per video line*** defined in the VESA specification for any specific VESA mode [1]. The counter operates at the speed of the pixel clock. It counts up, initially from zero, at every pixel clock until it hits $max\_pix - 1$, and then it will roll over and start again. The value stored at this counter is latched out by the active edge of the HS. Therefore, when HS is frequency locked to the pixel clock, the counter's value should be a **fixed** number. In other words, the counter should output the **same** value at each active edge of HS after the ADPLL is locked.

2) Implementation: The working mechanism of this phase detector can be explained as follows.

1) Detect the active edge of the incoming time base signal (HS) by latching out the counter's value at active edge of HS.
2) Record this value generated by above step (pix_cnt).
3) Convert this value to phase error by using the following formula:

$$\begin{aligned}&\text{If}\left(pix\_cnt < \frac{max\_pix}{2}\right)\\&\quad\{phase\_error = pix\_cnt + 1\}\\&\text{Else}\\&\quad\{phase\_error = pix\_cnt - max\_pix\}.\end{aligned}$$

3) Dead Zone: As addressed before, max_pix is the upper limit of this counter. The parameter pix_cnt is a pointer that can move within the range of $[0, max\_pix - 1]$. This pointer is used to indicate the location of the active edge. The phase detector is designed in such way that when HS and pixel clock are locked, the pointer should fix at value 0, which corresponds to angular phase difference of 0 radian or 2π radian. Ideally, when the two signals are locked, we hope that the detector output, $phase\_error$, is zero. But $phase\_error = 0$ will produce a large pointer (HS active edge) uncertainty, or "dead zone," since the pointer (pix_cnt) can be anywhere between $max\_pix - 1$ and 1 (with value 0 in middle) to satisfy this condition. This dead zone produces an overall system phase inaccuracy that is usually not acceptable. To reduce this dead zone, the $phase\_error$ is modified as shown in the above formula. The minimum phase error is either $+1$ or -1, but never zero. Since the phase detector's output in steady-state is either $+1$ or -1, the dead zone is effectively limited to the size of the sampling register's setup time.

4) Transfer Function: The transfer function of this phase detector is

$$\mathrm{PD}_{\mathrm{out}}(Z) = H_{\mathrm{pd}}(Z)^{*}\mathrm{PD}_{\mathrm{in}}(Z) = G_{\mathrm{pd}}^{*}\mathrm{PD}_{\mathrm{in}}(Z)$$

where G_{pd} is the phase detector gain.

Since the output of the phase detector is phase error, its angular phase range is $-\pi$ radian to π radian. Also, the detector's formula in Section II-B2 shows that its output is linear, with maximum outputs of $\pm max\_pix/2$. Thus, $G_{\mathrm{pd}} = max\_pix/(2\pi)$.

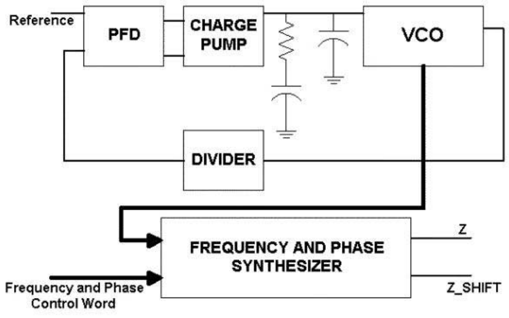

Reference
PFD
CHARGE PUMP
VCO
DIVIDER
FREQUENCY AND PHASE SYNTHESIZER
Frequency and Phase Control Word
Z
Z_SHIFT

Fig. 3. Structure of the DCO.

C. Digital Control Oscillator

In conventional PLL architecture, one of the key elements is the VCO. The VCO output frequency is a function of input voltage, which is proportional to the phase error signal. Therefore, the feedback mechanism can eventually eliminate the phase difference and make the VCO's output frequency/phase lock to the reference. In our ADPLL, the frequency generation mechanism is totally different due to the fact that our phase error signal is not a voltage of analog domain but a digital value. The counterpart of VCO used in this system is a DCO. This DCO will take a digital value as input and generate a signal whose frequency is related to this input value in certain way. The construction of this DCO is based on "flying-adder" frequency and phase synthesis architecture [2]–[4].

1) Structure of the DCO: Fig. 3 is the system block diagram of this DCO [2]. In this system, a very simple PLL/VCO is used to generate multiple phases (in this case, 31) that will be utilized by the flying-adder synthesizer to generate the desired frequency and phase. In normal operation, the PLL/VCO is running at a **fixed** frequency. Since no bandwidth requirement is specified for this PLL, the design can be focused on, instead of dynamic responds, stability and jitter immunity. Also, since the accuracy of the *R* and *C* components in this loop is not critical, they can be constructed by using transistors and poly resistors. This PLL is not qualified for the term "all-digital PLL" since the VCO's control signal is an analog voltage. Due to the unique feature of fixed frequency, this simple PLL can be designed easily with very low cost in terms of area and design complexity.

Fig. 4 is the circuit schematic of flying-adder synthesizer. In this circuitry, $\mathrm{FREQ}[32:0]$ is the 33-bit frequency control word, Z is the output signal with desired frequency, and $\mathrm{VCOOUT}[30:0]$ is the 31 reference signals from the analog VCO.

The principle idea of flying-adder architecture is to use multiple equally spaced phases generated from a VCO to synthesize various *frequency* and *phase*, by triggering the flip-flops at predestined time. In Fig. 4, there are two paths of logic cells of flip-flop, NAND, XOR/XNOR, register and adder. Each path also has one $32 \rightarrow 1$ multiplexer (MUX) that is used to select the tick from the VCO to trigger the flip-flop. Each path is responsible for generating the rising (or falling) edge of the synthesized output Z. The two paths are interlocked by the NAND gates so that only one path is active at any given time. The pipeline registers are used to ensure that all the adders have one full cycle

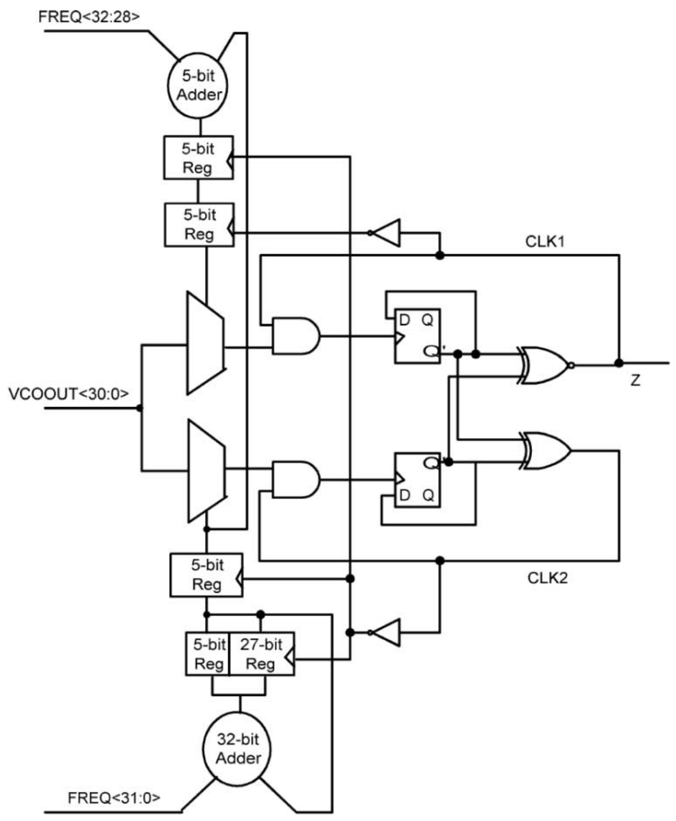


Fig. 4. Flying-adder frequency synthesizer.

time of Z to function. The output Z is also fed back as timing clock for the registers, to ensure the proper data flow. The detailed description of this circuitry can be found in [2].

2) Frequency Formula: For this DCO, the internal VCO is locked to a 14.31818-MHz crystal by a divider of eight. The VCO has 31 delay stages. Thus, the time difference between any two adjacent phases is 0.282 ns. The relationship between the control word and the output frequency can be expressed as [2]

$$\text{FREQ}[32{:}\,0] = \left(\frac{1}{f * 0.282}\right)$$

where f is the desired frequency in megahertz. FREQ$[32:0]$ is the frequency control word, FREQ$[32:27]$ is for the integer part, and FREQ$[26:0]$ is the fractional part. The fractional part is needed to generate certain “time-average” frequencies that cannot be produced otherwise. For example, if 160 MHz (6.25 ns) is the desired frequency, then

$$\begin{aligned}\text{FREQ}[32{:}\,0] &= \frac{6.25\text{ ns}}{0.282\text{ ns}}\\ &= 22.19381\\ &= 010\,110.001\,100\,011\,001\,110\,110\,001\,000\,001\text{b}.\end{aligned}$$

3) Transfer Function of DCO: The VCO is an integrator. In S-domain, its transfer function is $H(S) = G/S$, where G is the VCO gain. In Z-domain, the DCO can be modeled by

$$H(Z) = \frac{G_{\text{vco}}}{(1 - Z^1)}$$

through impulse-invariant Z-transform. In Fig. 2 of the ADPLL system, the frequency control word is initially set to a value that generates a pixel clock at a frequency that is specified in VESA specification. Then, based on the actual HS frequency, a phase error signal is generated. This error signal will be converted to an incremental change to the current frequency control word that will adjust the pixel clock’s frequency. The G_{vco} in this ADPLL is

$$G_{\text{vco}} = TC{*}2\pi{*}0.282$$

where TC is the number of pixels per video line for any specific VESA mode.

4) Use of Flying-Adder as DCO: The requirement for the timing generation block in Fig. 1 is to generate the pixel clock by locking to HS. Usually, HS is a low-frequency pulse and this pulse only presents in a very short time during one video line. If a conventional PLL is used for locking to this HS, then the reference signal is absent for most of time during the video line. This could cause pixel clock’s frequency, which is the HS frequency multiplied by the number of pixels per line, drifting away from the desired value. This drift will become more severe toward the end of the video line. In our ADPLL case, the frequency drift is nonexistent since the frequency control word is fixed during any one video line.

Unlike the analog VCO that requires certain number of cycles to lock to new frequency when the control signal is changed, the flying-adder synthesizer can adjust its output to new frequency in next cycle. This makes this architecture an excellent frequency source. Consequently, in our ADPLL application, the DCO responds instantly with input change.

Theoretically, any frequency within a certain range can be generated by the flying-adder architecture. In practice, the frequency resolution is related to the number of fractional bits [4]. In this application, there are 27 bits reserved for the fractional part, which is more than enough for generating all the frequencies required by this application.

Another unique feature of this DCO is that it can generate a different phase (delay) version of the pixel clock [2], which is extremely useful for precisely selecting the ADCs’ trigger time for better data quality.

D. Adaptive Loop Filter

The low-pass filter used in this ADPLL is implemented in software that is executed on an on-chip microprocessor shared with other functions. The microprocessor computes the frequency control word, as shown in Section II-C, for the DCO by using the output from the phase detector (through the low-pass filter). The signal flow chart of this software filter is shown in Fig. 5. Its transfer function is

$$H(Z) = G_1 + \frac{G_2}{1 - Z^{-1}}.$$

From the flow chart and the transfer function, it is clear that the implemented first-order filter is a simple integrator. If a phase error persists for an extended period of time, the output of the loop filter will ramp up or down depending on the direction and magnitude of the phase error. Proportional gain G_1 and integral gain G_2 determine the filter response. They can be adjusted

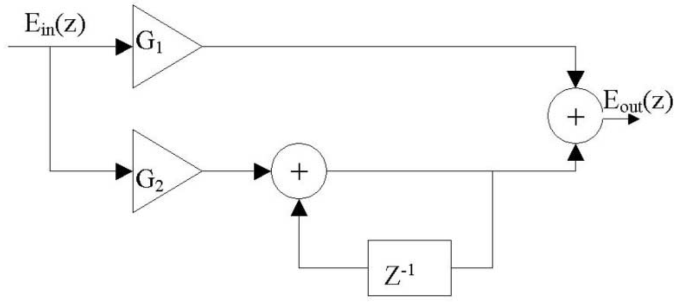


Fig. 5. Signal flow chart of the low-pass filter.

dynamically on the fly, with greater gain in the start-up process for fast locking and smaller gain in steady-state for better stability. This software filter is executed at the rate of HS and the response speed is sufficient for this application.

The advantages of using this software filter are: 1) no off-chip resistors and capacitors are needed; 2) the gain of G_1 and G_2 can be adjusted on the fly; and 3) this filter is process, temperature, and voltage independent.

III. Modeling of the ADPLL

A. Linear Second-Order System in S-Domain

Generally speaking, a PLL is a feedback control system. Assuming that the phase error is within a limited range, this feedback system can be further simplified as a linear feedback system. This assumption is reasonable for most applications since a real PLL has a locking range. Beyond the locking range, the PLL will become unstable. In other words, locking will not be guaranteed outside this range. Since the system is now described in continuous-time domain, the transfer function of each component in the loop is given in Laplace-transform format:

$$H1(S) = \frac{G_{lp}}{G_{lp+S}}$$
$$H2(S) = \frac{G_{\text{vco}}}{S}$$

where $H1$ and $H2$ are transfer functions of loop filter and VCO, respectively. Thus, the closed-loop transfer function is

$$H_{cl}(S) = \frac{G_{lp}G_{\text{vco}}}{S^2 + G_{lp}S + G_{lp}G_{\text{vco}}}.$$

This is a second-order system and all the existing second-order system analysis in automatic control system can be utilized. In control theory, the transfer function of standard second-order prototype system is [5]

$$H_s(S) = \frac{\omega_n^2}{S^2 + 2\zeta\omega_n S + \omega_n^2} = \frac{\omega_n^2}{(S - S_0)(S - S_1)}$$

where ω_n is defined as natural undamped frequency, and ζ is damping ratio. These two parameters are usually used to specify performance requirements of a system. Most transient responses can be determined based on these two parameters. In this case, these two parameters relate to G_{lp} and G_{vco} as

$$\omega_n = \sqrt{G_{lp}G_{\text{vco}}}$$
$$2\zeta\omega_n = G_{lp}$$

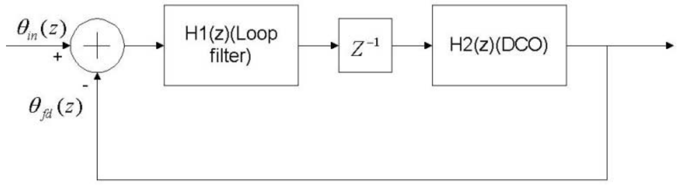


Fig. 6. Signal flow chart of sampled data system.

and the two poles are

$$S_0 = -\zeta\omega_n + j\omega_n\sqrt{1-\zeta^2}$$
$$S_1 = -\zeta\omega_n - j\omega_n\sqrt{1-\zeta^2}.$$

B. Second-Order System in Z-Domain

The above model can be directly applied to the PLL in continuous-time domain. But for systems based on sampled data, they should be modeled in discrete-time domain. Normally, the output responses of a discrete-time system are also functions of continuous-time variable t. Therefore, the goal is to map the system that meets the performance requirements specified by ζ, ω_n to a corresponding model in Z-domain. The signal flow chart of our second-order system in Z-domain is shown in Fig. 6.

In this chart, Z^{-1} is a unit delay that is used to represent a register. As shown in Sections II-C and D, the transfer functions of the loop filter and DCO can be expressed in the following generic terms:

$$H1(Z) = \frac{aZ - 1}{Z - 1}$$
$$H2(Z) = \frac{cZ}{Z - 1}.$$

By using these two functions, the closed-loop transfer function of Fig. 6 can be derived as

$$H(Z) = \frac{acZ - c}{Z^2 + (ac-2)Z + (1-c)} = \frac{N(Z)}{(Z - Z_0)(Z - Z_1)}.$$

C. Mapping From S-Domain to Z-Domain

By discrete-time transformation, two poles of this system in Z-domain can be mapped from the poles in S-domain as follows:

$$Z_0 = e^{S_0 T_s} = e^{\left(-\zeta\omega_n T_s + j\omega_n T_s\sqrt{1-\zeta^2}\right)}$$
$$Z_1 = e^{S_1 T_s} = e^{\left(-\zeta\omega_n T_s + j\omega_n T_s\sqrt{1-\zeta^2}\right)}$$

where T_s is the sampling period of the discrete system. Now the implementation procedure can be fully described as follows. The system requirement is first expressed in terms of ω_n and ζ as is usually done in continuous-time domain. Then, the requirement is mapped to the discrete system in the format of pole locations, which will determine the transient response of the ADPLL. The numerator of the transfer function can be a constant scaling factor, or zeros can be introduced to fine-tune the system performance. Section III-D will present a real example of this procedure.

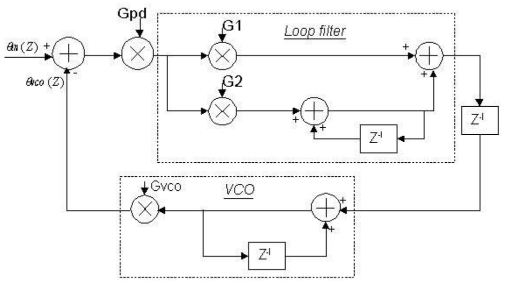


Fig. 7. Implementation of ADPLL.

D. One Case Study

The signal flow chart in Z-domain of this ADPLL is shown in Fig. 7, which is one implementation of the generic signal flow chart of Fig. 6.

The transfer functions of each component have already been described in Section II. The closed-loop transfer function then can be derived as

$$H(Z) = \frac{\theta_{\rm vco}(Z)}{\theta_{\rm in}(Z)} = \frac{(k_1+k_2)Z - k_1}{Z^2 + (k_1+k_2-2)Z + (1-k_1)}$$

where $k_1 = G_{\rm pd}G_{\rm vco}G_1$, $k_2 = G_{\rm pd}G_{\rm vco}G_2$.

By using the mapping approach developed in Section III-C, we get

$$\begin{aligned} k_1 &= 1 - e^{-2\zeta\omega_n T_s} \\ k_2 &= 1 + e^{-2\zeta\omega_n T_s} - 2e^{-\zeta\omega_n T_s}\cos\left(\omega_n T_s\sqrt{1-\zeta^2}\right). \end{aligned}$$

The design case that will be studied is an ADPLL that needs to recover the pixel clock from the PC graphic card's XVGA output. The HS frequency of XVGA with 75 frames/s refresh rate is $f_s = 60023$ Hz [1]. This is also our ADPLL's sampling rate, or $T_s = 0.01666$ ms. The number of pixels per video line for XVGA is 1312, or $T_s = 1312T_p$, where T_p is the pixel clock's period. The required locking time is $t_{\rm lock} <= 1$ ms. The requirement for the system dynamic response is that only one overshoot is allowed during the locking process for stable system response and reasonable settling time. Based on those system requirements, the following parameters can be determined:

$$\begin{aligned} \zeta &= 0.707 \text{ (to guarantee one overshoot only)} \\ \omega_n &= \frac{4}{(\zeta^* t_{\rm lock})} = 5657.71 \text{ rad/s [6]}. \end{aligned}$$

This will result in $k_1 = 0.1248$, $k_2 = 0.0083$. The loop filter gains G_1 and G_2 can be calculated accordingly since $G_{\rm pd}$ and $G_{\rm vco}$ are fixed and known. The resulting transfer function is

$$H(Z) = \frac{0.1331Z - 0.1248}{Z^2 - 1.867Z + 0.8752}.$$

The step responses of both $H(Z)$ and $H(S)$ of the second-order prototype system are shown in Fig. 8. It can be seen that, for the implemented system, lock time = 1.28 ms (versus 1 ms specified), maximum overshoot percentage = 16%. Also, it can be seen is that the two responses are noticeably different, as expected. Directly mapping from S-domain's prototype system to Z-domain and having exactly the same responses is theoretically possible, but the resulting system may not be implementable. The main goal of the mapping is to use the prototype system as guideline to construct an implementable system, with predictable performance. In our real ADPLL system, a zero has been introduced for implementation and performance issues.

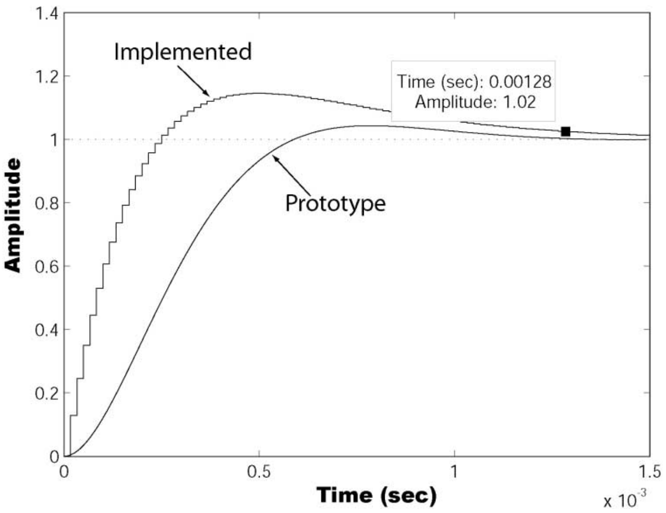


Fig. 8. Step response of the ADPLL.

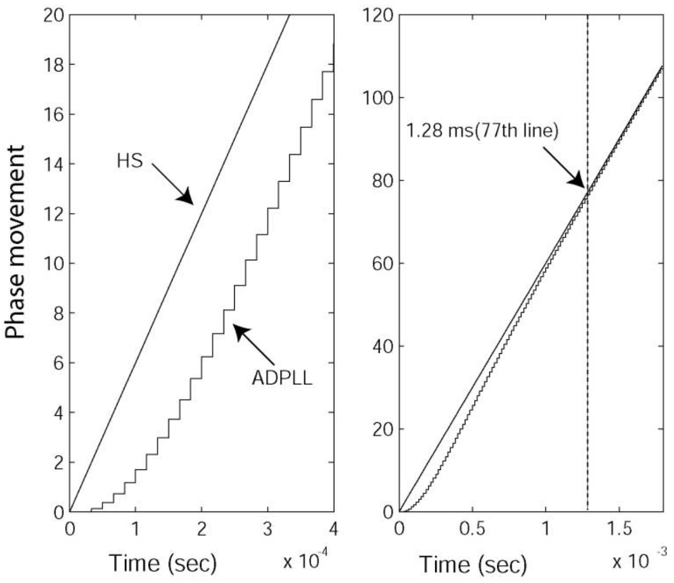


Fig. 9. Phase response to a frequency change.

Fig. 9 is the phase response of this ADPLL to a frequency change from 0 Hz to 60.023 kHz, which is the HS frequency. This simulation is the study of this ADPLL's dynamic behavior when it starts from initial reset state to the state of locking to the HS. The plot at left is the ADPLL's performance during the first 0.4 ms. The straight line is signal HS's phase movement as time passes. Its derivative is the HS frequency, or 60 kHz. The step size shown in the plot is the period of the HS, and is also the sampling period of this discrete system. The ADPLL's loop variables are updated at this rate. Each step corresponds to one video line. The plot at right is for simulation time of 1.8 ms. It shows that after 1.28 ms (about 77 video lines), the ADPLL is locked to the HS. Also shown is that both the phase and frequency steady-state errors of this system are zero, which will be proven theoretically in Section IV.

IV. Stability and Steady-State Error Analysis

A. Stability of the ADPLL

One mandatory requirement for designing any PLL is that the system must be stable. The stable condition for a discrete-time system is that the roots of the characteristic equation must be inside the unit circle $|Z| = 1$ in Z-plane. Normally, after a system is implemented, numerical coefficients can be substituted into the characteristic equation, which can then be solved numerically. The positions of the poles can be used to determine if the system is stable. However, in practice, this method is difficult to use since numerical coefficients usually are not available at the beginning of the process.

One of the most efficient criterions for testing the stability of a discrete-time system is Jury's stability criterion [5]. This method can guide the design of our ADPLL to converge to a stable system quickly, without performing a large amount of numerical calculation. For a second-order system, according to this criterion, the necessary and sufficient conditions for stability are

$$\begin{aligned} \Delta(1) &> 0 \\ \Delta(-1) &> 0 \\ |a_0| &< a_2 \end{aligned}$$

where Δ is the characteristic equation of the second-order system, or the denominator of the transfer function

$$\Delta(Z) = a_2 Z^2 + a_1 Z + a_0 = 0.$$

These conditions will guarantee that no roots are on or outside the unit circle. Applying these conditions to our transfer function of Section III-D, the stable condition for our ADPLL architecture can be derived as

$$\begin{aligned} 0 < k_1 < 2 \\ 0 < k_2 < 4. \end{aligned}$$

B. Steady-State Error of the ADPLL

Steady-state error analysis is another important aspect in designing a PLL system. This section will investigate the phase and frequency steady-state errors of this ADPLL.

1) Phase Error: Assuming that the phase of the input signal has a step change $\Delta\theta$. In time domain, this can be described by using the step function

$$\Theta_{\text{in}}(t) = \Delta\Theta \bullet u(t).$$

In Z-domain, this can be expressed as

$$\Theta_{\text{in}}(Z) = \frac{\Delta\Theta \bullet Z}{Z - 1}.$$

The phase error function $E(Z)$ can be derived as

$$E(Z) = [1 - H(Z)]\Theta_{\text{in}}(Z)$$

or by using our transfer function of Section III-B

$$E(Z) = \frac{\Delta\Theta Z(Z-1)}{Z^2 + (ac-2)Z + (1-c)}.$$

According to the Final-Value Theorem [5], the final value $e(kT)$, or steady-state error, in time domain is

$$\lim_{k\to\infty} e(kT) = \lim_{z\to 1}(1 - Z^{-1})E(Z).$$

By applying this theorem to $E(Z)$, it is clear that the steady-state phase error of this ADPLL is zero:

$$\lim_{k\to\infty} e(kT) = \lim_{z\to 1} \frac{\Delta\Theta(Z-1)^2}{Z^2 + (ac-2)Z + (1-c)} = 0.$$

2) Frequency Error: For a frequency jump from ω_0 to ω_1, or $\Delta\omega = \omega_1 - \omega_0$, it can be converted to phase change as

$$\Theta_{\text{in}}(t) = \Delta\omega \bullet t$$

or in Z-domain

$$\Theta_{\text{in}}(Z) = \frac{\Delta\omega T_s Z}{(Z-1)^2}$$

and

$$E(Z) = \frac{\Delta\omega T_s Z}{Z^2 + (ac-2)Z + (1-c)}.$$

By the same token, we can conclude that the frequency steady-state error of this ADPLL is also zero:

$$\begin{aligned} \lim_{k\to\infty} e(kT) &= \lim_{z\to 1}(1 - Z^{-1})E(Z) \\ &= \lim_{z\to 1} \frac{\Delta\omega T_s(Z-1)}{Z^2 + (ac-2)Z + (1-c)} = 0. \end{aligned}$$

V. ADPLL Implementation

A. Implementation Overview

This ADPLL is a system of digital functions (phase detector, counter, DCO), analog functions (VCO, charge pump, filter), and software function (loop filter). Structurally, it can be viewed as composed of two PLL loops. The internal loop of Fig. 3 is aimed at generating the multiple evenly spaced reference phases for the frequency synthesizer. As mentioned before, the design of this PLL/VCO is of little analog complexity since it is running at a fixed frequency. The accuracy of the *R* and *C* components of this PLL is not critical and can be easily implemented on-chip. Therefore, this analog function can be integrated with the rest of the digital functions in the same digital CMOS process without much difficulty. The digital functions of the phase detector, counter, and DCO are implemented as one physical block, in the fashion of RTL coding, synthesis, place and route in an ASIC library. The $32 \to 1$ MUXs in Fig. 4 are critical blocks in the DCO. Thus, they are designed and laid out manually with great care to ensure that all the paths from 32 inputs to output have the same delay and the address decoding is fast enough for proper operation. The loop filter is implemented in software with great flexibility for modifying the ADPLL's behavior on the fly. The on-chip microprocessor used is a 16-bit custom-built RISC machine of 30 MIPS. The majority of this processor's computation power is allocated to perform various other tasks on the chip. The ADPLL presents less than 5% load. This ADPLL is integrated in a graphic digitizer system that is implemented in a 0.6-μm

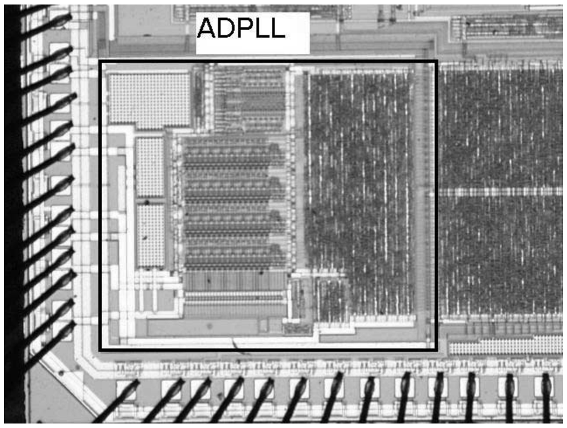


Fig. 10. Die photograph of the ADPLL.

CMOS process with 3.3-V supply. It is responsible for generating all pixel clocks within 10–80 MHz for various VESA video modes (the 80-MHz upper frequency is limited by the ADCs, not the ADPLL). The ADPLL consumes $\sim$180 mW by itself, with an area of $\sim$1.8 mm^2. A 14.31818-MHz crystal is used as a reference for the internal simple PLL. Fig. 10 shows the corner of the die where this ADPLL is located.

B. Analog PLL/VCO Design Considerations

Since there is no loop dynamic response requirement in this PLL's specification, the design can be oriented to stability and jitter immunity. In terms of phase noise (or jitter) performance, the PLL behaves like a low-pass filter to noise from the input reference. In this design, a narrow bandwidth is chosen for minimizing the reference noise and for a very stable loop.

A series of single-ended delay stages is chosen for the structure of VCO. This structure has poor power supply rejection ratio (PSRR) but has very little analog complexity. For phase noise injected into the VCO, the PLL loop behaves like a high-pass filter, which requires wide bandwidth to reduce its impact [7]. To solve this problem, the VCO gain in the neighborhood of the operation frequency is designed to be very small. Thus, it is less sensitive to the variation of power supply voltage and VCO control voltage.

C. VCO Layout Considerations

The VCO within the frequency synthesizer is 31 single-ended delay stages. One of the critical issues during the layout implementation is that the 31 VCO outputs have to be brought to the MUXs' inputs in equal lengths, or delays. Any mismatch will show up as deterministic jitter on the frequency synthesizer's output. Fig. 11(a) shows one VCO layout configuration (we use only seven stages for illustration purposes). This configuration will make the task of bringing the VCO outputs to the MUXs easy, but the wire connection from stage 7 to stage 1 is not equal to the connections of other stages and thus causes mismatch. Configuration of Fig. 11(b) will guarantee that all the connections within VCO are equal lengths, but will cause difficulty in bringing the outputs to MUXs. Fig. 11(c) is our solution, which ensures the equal-length connections within VCO and the equal-length layout wiring from VCO to MUXs.

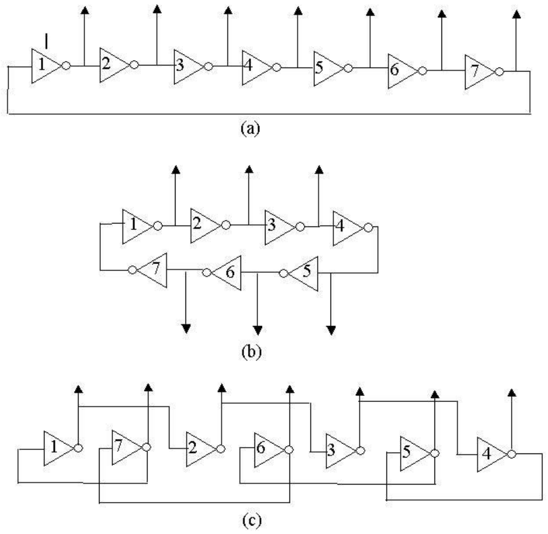


Fig. 11. Layout configurations of the VCO. (a) Chain. (b) Loop. (c) Coiled chain.

D. Simulation Strategy

1) Analog PLL: The so-called analog PLL that is responsible for generating the multiple references is designed and simulated in transistor level by using SPICE since all the functions in this loop are analog components.

2) Frequency Synthesizer: As shown in Fig. 4, it is clear that the flying-adder synthesizer is constructed by logic gates. A VHDL model is first developed and simulated in RTL level. Then the model is synthesized into logic gates and gate-level simulation is performed to verify the system functionality. Finally, after place and route, transistor-level SPICE netlist is generated with parasitic information. This SPICE netlist of the synthesizer is simulated intensively in order to evaluate the system performance. During these simulations, a SPICE MACRO model is used to represent the analog PLL/VCO since a real PLL transistor-level simulation is very time consuming, impractical, and unnecessary for this situation.

3) ADPLL System: The ADPLL system is first designed and simulated in Matlab. After loop performance has been determined and loop parameters have been finalized, VHDL RTL model is developed. Finally, after synthesis, when the gate-level netlist is available, intensive gate-level simulations are performed to verify the whole system's functionality. During these simulations, a VHDL model created in step 2) is used to represent the DCO.

E. Laboratory Test Result

This ADPLL has been tested in the laboratory for locking to HS signals of various VESA modes. Fig. 12 is one snapshot of digital scope. The signal shown on the image is the pixel clock of XVGA (1024×768) mode at 75-Hz refresh rate. As shown, a pixel clock of 78 MHz is generated by locking this ADPLL to HS. The storage feature of this digital scope is used to show the long-term jitter. The first rising edge of the signal is used as a reference. The second rising edge is the accumulation of many cycles. This result shows long-term jitter of $\sim$1.5 ns

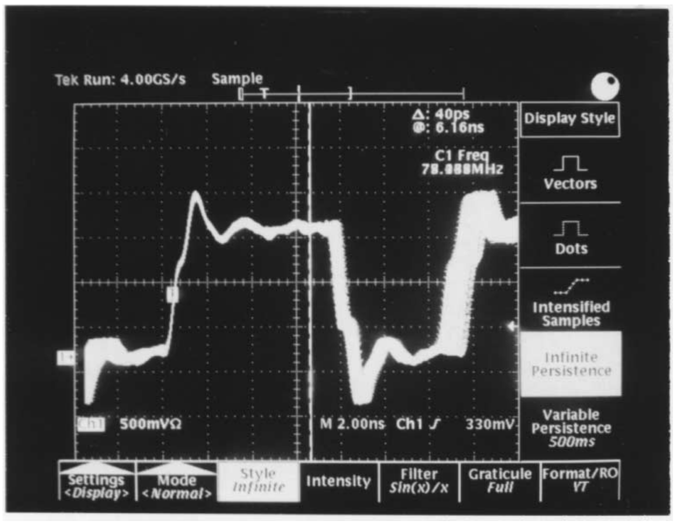


Fig. 12. Locking image.

peak-to-peak. The reasons for this size of jitter are: 1) the reference HS signal of 60 kHz is very noisy with significant jitter in itself; 2) the phase detector has a dead zone of a flip-flop's setup time, which is about ~500 ps in this ASIC library. This jitter performance is acceptable for this application since our ADPLL is used for the recovery of pixel clock from HS, not for high-speed data communication applications. For example, for XVGA at 75-Hz refresh rate, one pixel time is 12.8 ns. A worst case 1.5 ns jitter on pixel clock will not have a noticeable effect on the display, especially when we have the flexibility of adjusting the sampling phase.

VI. Conclusion

A hardware and software hybrid all-digital PLL was presented in this paper. The unique feature of this ADPLL is that it uses a flying-adder architecture based digital control oscillator to function as the VCO of conventional PLL. This feature enables the utilization of digital values, instead of analog voltage, as loop variables. The phase detector and loop filter are all designed to operate on digital values. Furthermore, the filter is implemented in software with great flexibility. For our application of recovering pixel clock from HS signal, this ADPLL is quite sufficient with several additional advantages.

Acknowledgment

The authors would like to thank L. Viswanathan, L. Taylor, and F. Tsay for their help on this project

References

[1] *VESA and Industry Standards and Guidelines for Computer Display Monitor Timing, Version 1.0, Reversion 0.8*, 1998.

[2] H. Mair and L. Xiu, "An architecture of high performance frequency and phase synthesis," *IEEE J. Solid-State Circuits*, vol. 35, pp. 835–846, June 2000.

[3] L. Xiu and Z. You, "An flying-adder architecture of frequency and phase synthesis with scalability," *IEEE Trans. VLSI Syst.*, vol. 10, pp. 637–649, Oct. 2002.

[4] L. Xiu and Z. You, "A new frequency synthesis method based on flying-adder architecture," *IEEE Trans. Circuits Syst. II*, vol. 50, pp. 130–134, Mar. 2003.

[5] B. C. Kuo, *Automatic Control System*, 7th ed. Upper Saddle River, NJ: Prentice-Hall, 1994.

[6] R. C. Drof and R. H. Bishop, *Modern Control System*, 9th ed. Upper Saddle River, NJ: Prentice-Hall, 2000.

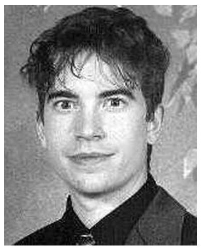
[7] W. F. Egan, *Phase-Lock Basic*. New York, NY: Wiley, 1998.

Liming Xiu (M'95–SM'03) received the B.S. and M.S. degrees in applied physics from Tsing Hua University, Beijing, China, in 1986 and 1988, respectively, and the M.S. degree in electrical engineering from Texas A&M University, College Station, in 1995.

He is currently a Design Engineer with Texas Instruments, Inc., Dallas, TX. He has worked on various mixed-signal devices, including video decoders, 3-D graphics controllers, and phase-locked loops. His research interests include digital and mixed-signal integrated circuits design and VLSI physical design. He has been elected as a Member of Group Technical Staff of Texas Instruments.

Wen Li (M'91) received the M.S.E.E. degree from the University of South Florida, Tampa, in 1991.

He has been with Texas Instruments, Inc., Dallas, TX, since 1996. Currently, he is a Design Engineer in the TI digital audio and video group. While with TI, he has worked on projects which have included video/image products and DSP coprocessors. His main research interests are in VLSI implementation of signal processing, especially in video and image processing. He was elected as a Member of Group Technical Staff of Texas Instruments in 1999. Before joining TI, he worked as a Design Engineer with Thomson Multimedia and with Vela Research.



Jason Meiners (M'96) received the B.S.E.E. degree from the University of Illinois at Urbana-Champaign in 1996.

He is a Design Section Manager and Member of Technical Staff with Texas Instruments, Inc., Dallas, TX. He leads the digital video design team for the High Performance Analog business unit. His specialty for the past five years has been digital video and graphics decoding, including real-time signal acquisition, genlocking, auto signal identification, and adaptive comb filtering. Prior to leading the video decoder team, Jason worked in the Multimedia group of the Mixed Signal Products business unit. His work there included phase-locked loops, high-speed bus interfaces, and graphics processors.

Rajitha Padakanti (M'99) was born in Hyderabad, India, in 1977. She received the B.Tech. degree from Osmania University, Hyderabad, in 1998 and the M.S degree in computer engineering from Iowa State University, Ames, in 2001. During her graduate studies, she was engaged in the research of data converters and phase-locked loops.

In 2001, she joined the Digital Video product division of Texas Instruments, Inc., Dallas, TX, as a Design Engineer. Since then, she has been involved in the design of low-power low-jitter phase-locked loops for video products. Her research interests include low-noise low-power phase-locked loops, algorithmic ADCs, and other mixed-signal blocks.