A “Flying-Adder” Architecture of Frequency and Phase Synthesis With Scalability

Liming Xiu, Member, IEEE, and Zhihong You, Member, IEEE

Abstract—Most of today’s digital designs, from small-scale digital block designs to system-on-chip (SoC) designs, are based on “synchronous” design principle. Clock is the most important issue in these designs. Frequency and phase synthesis is closely related to the clock generation. A frequency and phase synthesis technique based on phase-locked loop is proposed in [1] that delivers high performance, easy integration, and high stability. However, there are problems associated with this architecture, such as: 1) its highest deliverable frequency is limited by the speed of the accumulator and 2) the phase synthesis circuitry will not work well in certain ranges (dead zone) and in certain conditions (dual stability). This paper presents an improved architecture that addresses these problems. The new frequency synthesis circuitry has scalability for higher output frequency. It also has an internal node whose frequency is twice that of output signal. When duty cycle is not a concern, this signal can be used directly as clock source. The new phase synthesis circuitry is free of “dead zone” and “dual stability.” The improved architecture has better performance, is simpler to implement, and is easier to understand.

Index Terms—Clock generation, CMOS circuit design, frequency synthesis, phase-locked loop, phase synthesis, spread spectrum, voltage-controlled oscillator.

I. INTRODUCTION

The technique of frequency and phase synthesis has wide application in today’s consumer electronic and telecommunication systems. The architecture proposed by Mair and Xiu [1] has many unique features.

1) The synthesized frequency can be changed instantly in next cycle without any dynamic process found in traditional phase-locked-loop (PLL)-based technique.

2) Any frequency within a certain range can be achieved with controllable accuracy.

3) Various phase-shifted and various duty-cycle versions of the output signal can be generated.

4) The required voltage-controlled oscillator (VCO)/PLL is running at a single fixed frequency; therefore its design is much simplified.

5) The frequency and phase control words can be modulated easily to produce a highly accurate and predictable spread spectrum clock source, etc.

This architecture has been used widely in many designs since its invention and is commonly called “Flying Adder” architectures.

In the process of implementing this architecture into these designs, it is found that the architecture can be improved in several places to make it better and faster. The improvements are focused on the following areas:

1) making the frequency synthesis circuitry simpler and faster;

2) making the architecture of frequency synthesis scaleable for higher output frequency;

3) making the phase synthesis circuitry simpler and dead-zone, dual-stability free.

In this paper, Section II presents the problems of the current architecture. Section III is the improvement to the current frequency synthesis circuitry. The scalability of the new frequency synthesis architecture is discussed in Section IV. The improvement to the phase synthesis circuitry is described in Section V. Section VI is the implementation guideline. Section VII is an example of how this frequency synthesizer can be used as a digital controlled oscillator (DCO) in an all-digital PLL. Section VIII is the conclusion.

II. THE PROBLEMS OF CURRENT ARCHITECTURE

This section will be devoted to the discussion of the problems associated with current architecture.

A. The Problem of One-Path Frequency Synthesis Circuitry

Fig. 1 shows the principle idea of the Flying Adder frequency synthesis architecture. VCOOUT[31:0] is the 32 outputs from VCO. The 10-bit adder in the figure is responsible for generating the address for the 32–1 MUX, which will select one out of 32 available VCO outputs to trigger the D-flip-flop. The D-flip-flop is configured as a toggle flip-flop to generate the output frequency. This adder is responsible for both the rising and falling edges of the output Z. Therefore, the highest possible frequency of output Z is half the speed of this adder.

Theoretically, this mechanism is straightforward and works perfectly. In reality, there is a problem associated with the 32–1 MUX. As shown in Fig. 2, for any MUX other than 2–1 MUX, there is a potential glitch problem on the output when the address bits are switching. The physical reason for this glitch is that when multiple address bits are changed from one combination to another, there is no guarantee that all the individual bits can switch at the same time. This will result in some intermediate values in the process of switching. For example, in Fig. 2, the address sel[4:0] is switching from “0 0 0 0 0” to “1 1 1 1” at time t. The waveform Z is the ideal output. If for some reason we get an intermediate value of “1 0 1 0” during the process of switching, the actual waveform will be Z'. This potential glitch will falsely trigger the D-flip-flop and generate

Manuscript received July 26, 2001; revised January 8, 2002. Th authors are with Texas Instruments, Dallas, TX 75024 USA (e-mail: limingxii@ti.com; z-you@ti.com).
Digital Object Identifier 10.1109/TVLSI.2002.801607
wrong signal waveform. Hence, the circuitry of Fig. 1 cannot be used directly for frequency synthesis.

B. The Problems of Two-Path Frequency Synthesis Circuitry

The two-path frequency synthesis circuitry was shown in [1, Fig. 8], which is repeated here as Fig. 3. Although this circuitry works very well, it can still be improved in certain areas. One of the challenges of implementing this architecture is that the working mechanism is very hard to understand due to the interlocking of the two paths, the self-clocking of the registers, and the pipeline operation. This mechanism is inherently complex and requires designers’ attention.

1) Two-Path Interlocking: As explained in [1], this frequency synthesis circuitry is composed of two paths, which make the 32 VCO ticks look like 64 ticks. The two paths are interlocked through two AND gates and XOR/XNOR gates. The two AND gates and the feedback self-clocking of the registers ensure that at any given time, there is one path switched on (MUX output can be sensed by CLK pin of D-flip-flop) and one path switched off.

As addressed in Section II-A, if the output of MUX is directly connected to the clock pin of the D-flip-flop, a potential glitch can falsely trigger the flip-flop and generate the wrong frequency. This problem is solved elegantly through the use of interlock, as shown in Fig. 3 and demonstrated in Fig. 4. When the address bits of any of the two MUXs are switching, the output of that MUX is locked by the AND gate through the feedback. When the AND is in unlocked state, the mechanism of registers self-clocking ensures that the address bits are already stable. Therefore, the two D-flip-flops will never see any
In Fig. 4, “ANDx locked” means that the feedback input of the AND gates is zero. “ANDx unlocked” is for that input at “1.” “MUXx decoding” means that the address bits of MUX are switching; “MUXx stable” is for address bits in steady state.

This configuration of two D-flip-flops and XOR/XNOR gates works well. But it prevents this architecture from being able to be expanded to four paths (or even more) to increase the highest possible output frequency.

2) Gates’ Delay: One of the speed-limiting factors of this circuitry is the delays of AND, D-flip-flop, and XOR/XNOR [1]. Reducing or eliminating some of them will potentially make the output frequency higher.

3) The Speed of the Accumulator: As addressed in [1, Section V], the speed of the big accumulator is the bottleneck for higher output frequency. Differential VCO can deliver an even number of outputs (ticks) for the synthesizer. This can help to eliminate the need for modulus 31 check and increase the speed of accumulator significantly.

C. Problems Associated With Phase Synthesis

Fig. 5 shows the principle idea of phase synthesis. The adder in PHASE_GEN is used to add PHASE[4:0] to the current address of the MUX in FREQ_GEN to generate the proper delay signal Z_SHIFT. Since a finite time delay is associated with any physical adder, certain phase range cannot be achieved in this configuration (dead zone). The dead zone corresponds to the low value range of PHASE[4:0]. It is the time needed for the adder to finish its addition. One solution to this problem is to always use the second half (high value) of PHASE[4:0] and to invert the Z_SHIFT when the first half is needed.

When the two-path architecture is used for phase synthesis, as shown in [1, Fig. 12], there is an additional problem of “dual stability” when PHASE[4:0] and FREQ[32:0] are in certain combination. Under these conditions, there are two possible Z_SHIFT locations for a given PHASE[4:0] value [1, Fig. 11]. This phenomenon is called “dual stability” and is not acceptable for real application. The design tradeoff for compensating this problem is very complicated.

III. THE NEW FREQUENCY SYNTHESIS ARCHITECTURE

This section will discuss the techniques to improve the frequency synthesis circuitry of Fig. 3.

A. VCO Design

As shown in [1, Figs. 9 and 10], the accumulator has to perform the task of addition as well as modular 31 check. This modular check is needed because the VCO architecture of that design is an invert ring and the number of invert stages in the ring has to be an odd number. A differential VCO architecture as shown in Fig. 6 can be used to deliver an even number of outputs [2]–[4]. This will eliminate the need for modular 31 check and speed up the accumulator. In this design, a crystal of 14.31818 MHz is used as a reference clock. A divider of 20 is inserted in the PLL’s feedback loop. Therefore, the VCO is running at 286.3636 MHz (3.492 ns). The VCO has 16 differential delay stages with 32 outputs. The delay between any two adjacent outputs is $\Delta = 3.492\text{ ns}/32 = 109.1\text{ ps}$.

B. The New Synthesizer

As addressed in Section II-B, if we reduce or eliminate the delays of AND gates, D-flip-flops, and XOR/XNOR gates (t4, t5, and t6), we can potentially make the synthesizer faster. Fig. 7 is the improved architecture.

Comparing to Fig. 3, it can be seen that the two AND gates, two D-flip-flops, and XOR/XNOR gates have been replaced by one $2^{1} > 1$ MUX and one D-flip-flop. This modification achieves the same two-path interlock function but reduces the $t4 + t5 + t6$ significantly.
Another distinguishing feature of this modification is that the frequency of signal `TRIGGER`, which is the output of the `2^>1 MUX`, is twice the synthesized frequency `CLK1/CLK2`. Therefore, this new architecture not only improves the speed of the synthesizer but also provides a signal that is twice as fast as the synthesized output. The only disadvantage is that the duty cycle of `TRIGGER` is related to the `FREQ` word and is not controllable. In many situations, users do not care about the duty cycle of the clock signal, and for those applications, `TRIGGER` can be used directly as the clock.

It should be mentioned that the frequency calculation formula [1, (3)] is still valid. To the user of this frequency synthesizer, there is no difference.

C. Simulation Results

For this new design, a 0.13-μm 1.5-V CMOS technology is used. Fig. 8 is the collection of SPICE simulation results of frequency control word `FREQ[32:0]` versus synthesized frequency. `FREQ[32:27]` is the integer part; `FREQ[26:0]` is the fractional part. In Fig. 8, all the bits in `FREQ[26:0]` are set to zero; `FREQ[32:27]` is swept from “0×3F” to “0×0D.” The data points are the simulated output frequencies, each point corresponding to a `FREQ[32:27]` setting. All the data points in the plot are exact frequencies, not time-average frequencies. In other words, when `FREQ[26:0] = “0×000000”`, the synthesized frequencies do not contain any theoretical cycle-to-cycle jitter. The frequencies between these data points can be achieved with the help of `FREQ[26:0]`.

Fig. 9 shows the SPICE simulation of `FREQ[32:27] = “0×0D”`. The calculated output frequency should be $13 \times \Delta = 1,418.3$ MHz. The result from SPICE simulation is also 1,418 ns. Since `CLK1` has to drive 42 flip-flops (no clock tree in this design), its rising/falling edge is slower than that of `CLK1`, which drives only five flip-flops. `MUXOUT_UP` and `MUXOUT_LOW` are the outputs of the two `2^>1 MUXs`. `TRIGGER` is the output of the `2^>1 MUX`, which drives the key D-flip-flop. It can be seen that the frequency of `TRIGGER` is twice that of `CLK1/CLK2`. Theoretically, it can be proved that the synthesized frequency `CLK1/CLK2` will always be one-half of the VCO frequency, 143.1818 MHz in our case, when the synthesizer is disabled, or $EN = \emptyset$. This can also be seen in Fig. 9.

SPICE simulation suggests that the highest `CLK1/CLK2` frequency is 705 MHz for this process in weak condition. The limiting factor is the speed of the accumulator.

IV. THE SCALABILITY OF THE IMPROVED FREQUENCY SYNTHESIS ARCHITECTURE

As described in [1], the main advantage of circuitry in Fig. 3 over the circuitry in Fig. 1 is that the highest synthesizable output frequency of Fig. 3 is twice that of Fig. 1 because of the utilization of two paths. If four paths (or even more paths) can be used for this purpose, then the highest possible output frequency will be even higher. In other words, the architecture of scalability has the flexibility for expansion.

A. The Speed of Adders

Fig. 10 shows the relationship between the speed of the adders and the frequency of the output signal in a two-path configura-
Edges A and C complete a full cycle; edge B is the falling edge in between. Starting from edge A, adder1 is responsible for edge C; adder2 is for edge B. Therefore, the highest possible output frequency is limited by the speed of adder1. Adder1 is the big accumulator with fractional part, which is the 10-bit adder in PATH_A of Fig. 3. Adder2 is an integer adder that is the 5-bit adder in PATH_B. Adder2 is not the limiting factor for two reasons: 1) its size is much smaller than adder1 and 2) it also has a full cycle of time to finish addition because of pipeline operation.

Fig. 11 is the idea of utilizing four paths to improve the output frequency. Edges A and C complete a full cycle; A and E complete two cycles. Edges B and D are the falling edges in between. It can be seen that adder1, which is an accumulator with fractional part, is responsible for generating edge E, not C. It has two cycles of time to finish addition. Thus, the highest possible output frequency will be twice that of the two-path architecture if all other conditions are the same. Adders 2–4 are all integer adders, responsible for edges B–D, respectively. They all have plenty of time to finish their work because of the pipeline.

B. The Four-Path Architecture

As mentioned in Section III-B, the scalability of this architecture is made possible by the improvement of Fig. 7: all paths share the same D-flip-flop to generate the waveform. Fig. 12 is the schematic of a four-path frequency synthesizer.

In this configuration, there are four adders. Adder1 is an accumulator with fractional part, 5-bit integer, and 27-bit fraction. Adders 2–4 are all 5-bit integer adders. Starting from any given rising/falling edge of output Z, adder1 is responsible for the rising/falling edge two cycles downstream; adder3 is for the rising/falling edge one cycle downstream. Adder2 and adder4 are for the falling/rising edges in between. The inputs of all four 32−1 MUXs are connected to the 32 VCO outputs. The four outputs of these MUXs are connected to a 4−1 MUX whose output is used to trigger the D-flip-flop. The pipelined
registers are all clocked by proper clock signals to ensure that all four adders have two cycles of time to finish addition. The CLK_CNTL block is used to generate the various clock signals for registers and the select signals for the 4:1 MUX.

Since four paths are used in this circuitry, the 32 VCO outputs act like $4^3 \times 2 = 128$ ticks. Thus, seven bits are needed to represent these ticks. In Fig. 12, $FREQ[33:27]$ is the integer part and $FREQ[26:0]$ represents fractional part.

C. The CLK_CNTL Block

This block needs to perform two functions: 1) generate the clock signals $CLK1, 2, 3,$ and $4$ and 2) generate the select signals.
The most challenging task for this block is that it has to guarantee that the output of the 4 → 1 MUX is glitch-free. The design of this block is described below.

1) Gray Coding of sel5[1:0]: To ensure that the 4 → 1 MUX does not generate glitch by itself, CLK_CNTL will use Gray-coding style in its output sel5[1:0]. That is, at any given time, there is at most one bit switching among sel5[1:0]. The switching sequence used is 00 → 01 → 11 → 10 → 00. The reason that we can use Gary coding here but not in Section II-A is that the selection sequence MUX1 → MUX2 → MUX3 → MUX4 → MUX1 is known and fixed.

2) Proper Order of Clock Signals: To ensure that the glitches on the inputs of the 4 → 1 MUX, which are the outputs of the four 32 → 1 MXUs, not be passed to the output, the CLK_CNTL has to generate the sel5[1:0] and CLK1, 2, 3, 4 in proper order, as shown in Fig. 13.

As shown in Fig. 12, whenever there is a rising edge on signal TRIGGER, the CLK_CNTL will be triggered, and the sel5[1:0] and the waveforms of CLK1, 2, 3, 4 will be updated. For example, at the sixth rising edge of TRIGGER (Fig. 13), MUX2 (sel5[1:0] = 01) will be selected. But CLK2’s rising edge was generated one tick before, at the fifth rising edge of TRIGGER. This means that register REG2 was triggered at that time and the output of MUX2 should be stable at current time. This mechanism will ensure that all the possible glitches on the inputs of MUX5 are blocked.

CLK_CNTL is basically a state machine that can be constructed by the method of HDL coding and synthesis. This solution will guarantee that signal TRIGGER is glitch-free and the D-flip-flop will never be falsely triggered.

D. The Frequency Calculation Formula

Since we are using four paths to generate the desired frequency, a new formula has to be developed. If the time difference between two adjacent VCO ticks is Δ and the frequency control word is FREQ, then according to Fig. 11, we have the following equation:

$$2^*T = FREQ^*\Delta$$

$$FREQ = \frac{2^*T}{\Delta}$$

where T is the period of the synthesized frequency.

E. Cycle-to-Cycle Jitter

The maximum cycle-to-cycle jitters of one-path and two-path configurations are all Δ [1]. For the four-path case, the following study will show that the maximum cycle-to-cycle jitter is still Δ.

From Fig. 11, it can be seen that edges A and C complete one cycle, and edges C and E form another cycle. If Δt_{ac} is the time difference between A and C, Δt_{ce} is the difference between C and E; then the cycle-to-cycle jitter will be $J_{peak} = \text{abs}(\Delta t_{ac} - \Delta t_{ce})$.

As mentioned above, the integer part is FREQ[33:27]. The input to adder3, which is responsible for edge C, is (FREQ[33:28] + FREQ[27]), as shown in Fig. 12.

1) $FREQ[27] = 0$ (integer part is an even number).

If $FREQ[27] = 0$, then

$$\Delta t_{ac} = (FREQ[33:28] + FREQ[27])*\Delta$$

$$= FREQ[33:28]^*\Delta$$

$$= \left(\frac{FREQ[33:27]}{2}\right)^*\Delta.$$ 

And

$$\Delta t_{ce} = (FREQ[33:27] - FREQ[33:28])*\Delta$$

$$= \left(\frac{FREQ[33:27]}{2}\right)^*\Delta.$$ 

If no fractional part overflow happens in edge E, then $J_{peak} = \text{abs}(\Delta t_{ac} - \Delta t_{ce}) = 0$. If there is overflow, then $\Delta t_{ce} = (FREQ[33:27]/2 + 1)^*\Delta$ and $J_{peak} = \text{abs}(\Delta t_{ac} - \Delta t_{ce}) = \Delta$.

2) $FREQ[27] = 1$ (integer part is an odd number).

If $FREQ[27] = 1$, then

$$\Delta t_{ac} = (FREQ[33:28] + FREQ[27])*\Delta$$

$$= (FREQ[33:28] + 1)^*\Delta$$

$$= \left(\frac{FREQ[33:27] + 1}{2}\right)^*\Delta.$$ 

And

$$\Delta t_{ce} = (FREQ[33:27] - FREQ[33:28] + 1)^*\Delta$$

$$= \left(\frac{FREQ[33:27]}{2}\right)^*\Delta.$$ 

If no fractional part overflow happens in edge E, then $J_{peak} = \text{abs}(\Delta t_{ac} - \Delta t_{ce}) = \Delta$. If there is overflow, then $\Delta t_{ce} = (FREQ[33:27]/2 + 1)^*\Delta$ and $J_{peak} = \text{abs}(\Delta t_{ac} - \Delta t_{ce}) = 0$. 

Authorized licensed use limited to: TEXAS INSTRUMENTS VIRTUAL LIBRARY. Downloaded on April 9, 2009 at 23:06 from IEEE Xplore. Restrictions apply.
In all cases, the maximum cycle-to-cycle jitter will remain as $\Delta$. There is no possibility for cycle-to-cycle jitter’s being $2\Delta$, or greater.

If the fractional part is not used ($FREQ[26:0] = 0 \times 00000000$ all the time) and $FREQ[33:27]$ is only used as even number ($FREQ[27] = 0$ all the time), then this architecture will not produce any cycle-to-cycle jitter. In those cases, the synthesized frequencies are exact frequencies, not time-average frequencies.

**F. Simulation Results**

This four-path architecture is implemented in the same 0.13-$\mu$m technology. The adders are built at the same speed as adders in two-path architecture. Fig. 14 is the collection of simulation data of $FREQ[33:27]$ versus synthesized frequencies.

Fig. 15 shows the SPICE waveforms of $FREQ[33:27]$ versus synthesized frequencies. The calculated frequency is $763.83$ MHz. The SPICE output frequency is $763.53$ MHz. It can be seen that the signals of $CLK1, 2, 3, 4$ and $sel5[1:0]$ behave just as described in Fig. 13. The rising/falling edge of $CLK1$ is much slower than that of $CLK2, 3, 4$ since $CLK1$ has to drive 52 flip-flops. The others only need to drive five flip-flops.

Based on SPICE simulation, the highest synthesizable frequency of this circuitry is $916$ MHz in weak condition. The limiting factor in this case is block $CLK_CNTL$, which is working at the speed of $TRIGGER (2 \times 916$ MHz$)$. The adders only need to work at $916$ MHz/2 = $458$ MHz. The goal of this architecture scalability is to relax the requirement on the speed of adders. The new speed bottleneck is the block $CLK_CNTL$. This $CLK_CNTL$ can be designed much faster than adders since the logic in this block is much simpler. But since this block is working at twice the speed of output signal, the full potential of scalability is not achieved.

**V. THE NEW PHASE SYNTHESIS ARCHITECTURE**

As addressed in Section II-C, the phase synthesis architecture presented in [1] has some shortcomings. This section will provide one solution to solve those problems.

One of the many features of this “Flying Adder” synthesizer of Fig. 7 is that it can start from any initial conditions. No matter what the two 32 $\times$ 1 MUXs’ initial addresses are, the circuitry will produce the right frequency, after a few cycles, when the adders are enabled to work. This is very good for frequency synthesis, but it presents challenges if another signal of the same frequency but different phase needs to be generated from the original signal.
For the purpose of generating a phase-shift (or delayed) version of output, one solution is to modify the circuitry of Fig. 7 so that it always starts from a known state when EN is switched from zero to one.

Fig. 16 is the modified version of Fig. 7. Enable circuitries have been inserted between the pipelined registers in both paths. This will ensure that the address values of the two MUXs will stay at INIT1 when EN1 = '0'. EN1 is the latched version of EN by CLK2 so that when EN is switched from zero to one, the upper 32 >1 MUXs’ address will be changed from INIT1 to other value first. The circuitry of Fig. 16 generates the output Z with desired frequency. The one in Fig. 17 will generate output Z_SHIFT with the same frequency and desired phase (delay).
Fig. 16. The circuitry for generating \( Z \).

Fig. 17 is almost identical to Fig. 16 structurally. The only difference is the data input of the D-flip-flop. When \( EN2 = 0 \), the data input to the D-flip-flop is \( CLK1(Z) \) (from Fig. 16); when \( EN2 = 1 \), this D-flip-flop is configured as a toggle flip-flop. \( EN2 \) is the latched version of \( EN1 \) by \( CLK2P \). This design guarantees that the two D-flip-flops are always synchronous in both its clock and data pins so that \( Z \_SHIFT \) is synchronous with \( Z \) all the time. The time difference \( PHASE[5:0]*\Delta = (INIT2-INIT1)*\Delta \) is the desired delay. This solution eliminates the dead-zone problem of previous architecture. It also eliminates the dual-stability problem since only one \( Z \_SHIFT \) location exists for any given \( PHASE[5:0] \) regardless of \( FREQ[32:0] \).

The only drawback of this new phase synthesis architecture is that enable signal \( EN \) has to be set to zero for several cycles if \( PHASE[5:0] \) needs to be changed. This means that the frequencies of both \( Z \) and \( Z \_SHIFT \) will be shifted from the required to half of the VCO frequency for a short period of time (during \( EN = 0 \)) if phase delay of \( Z \_SHIFT \) needs to be adjusted. This has some negative impact if this circuitry is used in a feedback system for phase synthesis.

Fig. 18 is the SPICE simulation of this phase architecture. \( FREQ[32:27] = "0\times16" = 22 \ (417 \text{ MHz}) \) and \( FREQ[26:0] = "0\times000000" \). \( PHASE[5:0] = "0\times0A" = 10 \ (INIT1[5:0] = "0\times00", INIT2[5:0] = "0\times0A", \text{ delay of} \ 1.091 \text{ ns}) \). This figure clearly shows the delay of 1.091 ns between \( Z \) and \( Z \_SHIFT \). It also shows the \( EN, EN1 \), and \( EN2 \).

VI. THE IMPLEMENTATION GUIDELINE

To successfully implement this architecture to any real project, a great amount of detail has to be worked out. The design task can be partitioned into several subtasks.
A. PLL Design

One of the goals of this architecture is to reduce or eliminate the tough challenges associated with designing a high-performance PLL. The PLL used in this architecture is a very simple one. The only requirement is to provide $N$ stable reference signals of fixed frequency to the synthesizer. The VCO gain should be designed relatively small to make it less sensitive to noise on the control voltage. Since there is no requirement on dynamic response of this PLL, the design can be focused on high loop stability, not on wide loop bandwidth.

B. Synthesizer Design

The design of this frequency and phase synthesizer is mainly in digital domain. Other than the MUXs and the toggling D-flip-flop, the rest of the synthesizer can be designed in HDL coding and synthesis style, using standard application-specific integrated-circuit (ASIC) cells. The following steps are usually required.

1) Register-Transfer-Level (RTL) Development: During the development phase of this synthesizer, a logic simulator can greatly help the designer to understand the working mechanism of this architecture. By using a logic simulator, all the internal signals, especially the address of those MUXs, can be monitored constantly during the simulation. The simulator can be helpful in verifying that the RTL code is functionally correct before the next step is performed.

2) Logic Synthesis: After RTL code is verified, the next step is logic synthesis, to convert the HDL code to a gate-level netlist.

3) Gate-Level Simulation: The gate-level netlist from step 2 has to be verified again by using a logic simulator to ensure that it agrees with the RTL code. Then the netlist can be passed to a place and route tool for layout.
4) Performance Evaluation: To evaluate the performance of a particular design, the circuitry has to be simulated by a transistor-level simulator, such as SPICE. The gate-level netlist of step 2 can be converted to a SPICE netlist, with the parasitic information from layout back-annotated. SPICE simulation will report the speed and many other important characteristics of the design.

VII. THE FREQUENCY SYNTHESIZER AS DIGITAL CONTROLLED OSCILLATOR

All PLLs can be classified as three types: linear (LPLL), classical digital (DPLL), and all-digital (ADPLL). All blocks of LPLL (phase detector, loop filter, and VCO) are analog components. For DPLL, the only part that is really digital is the
phase detector; the loop filter and VCO of DPLL are still analog. In contrast to DPLL, ADPLL is an entirely digital system. The “digital” in ADPLL means that all the function blocks of the system are implemented by pure digital circuits and the signals within the system are digital logic values, not voltages. 

The digital counterpart of VCO is the DCO. Nowadays, the most frequently used DCOs are $\pm N$ counter DCO, increment–decrement (ID) counter DCO, and waveform synthesizer DCO [5]. The $\pm N$ DCO is simply a divider that divides the frequency of a reference signal, from a fixed high-frequency oscillator, down by $N$. The ID counter DCO is intended to operate in conjunction with those loop filters that generate CARRY and BORROW pulses. This type of DCO needs an ID clock, its output frequency depends on the CARRY and BORROW pulses. The waveform synthesizer DCO generates sine and/or cosine waveforms by looking up tables stored in read-only memory. It calculates the samples of the synthesized signal at fixed clock rate.

Since its output frequency is solely depended on the input control word, the Flying-Adder frequency synthesizer can function as a perfect DCO. The modulation bandwidth of this DCO is infinite since its output frequency can be changed instantly in next cycle. An example of application is presented in [6]. Fig. 19 is the block diagram of this second-order ADPLL.

In this implementation, the phase detector generates a phase error signal based on the phase difference of $\theta_\text{in}$ and $\theta_\text{VCO}$. The loop filter will determine the loop stability and steady-state error. Since the transfer function of analog VCO is $H(s) = G/s$, the DCO can then be modeled by $\hat{H}(z) = G_\text{VCO}/(1 - z^{-1})$ through impulse-invariant $z$-transform. By utilizing this frequency synthesizer as DCO, this ADPLL achieved the desired functionality with additional advantages, such as no passive R and C components, dynamical control of the loop gain on the fly, and easy implementation through standard cells.

VIII. CONCLUSION

Since its invention, this Flying-Adder architecture has been widely used in many projects within the author’s organization. Extensive study has been performed on this frequency and phase synthesizer, and it is found that the original circuitry can be improved in several areas. This paper presented the details of the new architecture. The following is a summary of the improvements:

1) The frequency synthesizer circuitry is much simplified and is optimized for speed.
2) The new frequency synthesizer circuitry has an internal signal that is twice as fast as the output signal. In many applications, this signal can be used directly as the clock.  
3) The new frequency synthesizer circuitry has scalability and can be expanded to multiple paths for higher frequency.
4) A new phase synthesis architecture has been proposed. The new circuitry is free of “dead zone” and “dual stability.”
5) Both the new frequency and phase synthesizer circuitries are much easier to be understood.

ACKNOWLEDGMENT

The authors thank G. Manganaro, R. Anderson, R. Padakanti, and W. Li for their help on this project.

REFERENCES


Liming Xiu (M’95) received the B.S. and M.S. degrees in applied physics from Tsing Hua University, Beijing, China, in 1986 and 1988, respectively. He received the M.S. degree in electrical engineering from Texas A&M University, College Station, and 1995. He is currently a Design Engineer at Texas Instruments, Inc., Dallas. He has worked on various mixed signal devices, including video decoders, three-dimensional graphics controllers, and phase-lock loops. His interests include digital and mixed-signal integrated circuits design as well as mixed-signal integrated circuits design.

Zhihong You, (M’95) received the B.S. and M.S. degrees from the Precision Instrumentation Department, Tsing Hua University, Beijing, China, in 1987 and 1989, respectively. She received the M.S. degree in electrical engineering from Texas A&M University, College Station, in 1993. She is currently a Senior Design Engineer at Texas Instruments, Inc., Dallas. She has worked on various mixed signal devices, including hard disk drive, medical instruments, etc. Her prime interest is analog VLSI circuit design. She also has strong working experience on digital VLSI circuit design.