A Fast and Power–Area-Efficient Accumulator for Flying-Adder Frequency Synthesizer

Liming Xiu, Senior Member, IEEE

Abstract—The Flying-Adder frequency synthesis architecture is a novel approach of generating frequencies on chip. Since its invention, it has been used in many commercial products to cope with difficult frequency generation challenges. Along the course of this architecture’s evolution, various circuit- and system-level problems have been resolved. In this paper, one remaining problem related to circuit implementation, namely, the construction of the accumulator, is studied. A new scheme is proposed to achieve the Flying-Adder accumulation function that not only has speed advantage but also is power and area efficient. The issue related to time-average frequency and jitter is also discussed.

Index Terms—Accumulator, adder, clock generation, Flying-Adder, frequency synthesis, phase-locked loop (PLL).

I. INTRODUCTION

The FLYING-ADDER frequency synthesis architecture is a new way of generating frequency on chip. Mair and Xiu [1] is the proof of concept of this architecture, and [2] and [3] are the circuit-level improvements. The Flying-Adder technique has two working modes: integer flying adder and fixed-voltage-controlled-oscillator (VCO) flying adder. The integer-Flying-Adder architecture is a powerful extension of the integer-N phase-locked-loop (PLL) architecture. It enhances the integer-N PLL solution space since it utilizes a phase divider that can reach a finer frequency resolution than a frequency divider can [4]. The fixed-VCO flying adder, on the other hand, is based on the rigorously formed concept of time average frequency [5], [6]. It can virtually achieve any frequency desired. Furthermore, with fixed-VCO-Flying-Adder PLL, the output frequency can be switched instantly from one frequency to another. The frequency response is instantaneous since the frequency generation circuit is outside the PLL. Since its invention, the Flying-Adder-architecture-based PLL (FAPLL) has been used in many commercial projects to solve problems that cannot be dealt with easily by conventional PLLs. Xiu [7] demonstrates how the Flying-Adder-based phase divider works harmonically with the frequency divider to generate many frequencies on chip to support the various functions of a complex system-on-a-chip. The work of Xiu [8] is the detailed example of how the Flying-Adder-based PLL can function as a digitally controlled crystal oscillator (DCXO) to save cost for an MPEG2 decoder chip. In [9], a Flying-Adder PLL is used as a digitally controlled oscillator to function as the VCO (DCO) in an all-digital PLL for a graphic digitizer chip. The work of Xiu et al. [10] is the case where FAPLL can be used to reduce the size of on-chip memory. The study of Xiu [11] is the summary of FAPLL applications in commercial products with several additional examples.

One of the key circuit components used in the Flying-Adder architecture is the adder/accumulator. It is the fixed-VCO-Flying-Adder-architecture speed bottleneck for two reasons: 1) The adder/accumulator needs to operate at the speed of the output frequency, and 2) the size of the adder/accumulator needs to be large for fine frequency resolution. These two requirements, namely, high speed and large size, make the implementation of the adder/accumulator a nontrivial task. It has been the most influencing limiting factor for achieving higher output frequency for the fixed-VCO-Flying-Adder architecture. This issue has been discussed intensively in [2].

Historically, the VLSI implementation of an adder has received substantial attention from researchers since addition is by far the most frequently used operation in circuit systems. For example, a digital signal processor relies heavily on the efficient implementation of arithmetic circuits to execute dedicated algorithms, such as convolution, correlation, and digital filtering. As the processor’s bus width enlarges and the processor’s speed increases, the VLSI implementation of an adder, which is the core element of complex arithmetic circuits, becomes increasingly difficult.

There are several types of parallel adder architectures available to meet various performance and resource requirements, such as ripple-carry adder (RCA), Manchester carry-chain adder, carry-skip adder, carry-select adder, carry-look-ahead adder, and carry-save adder [12]. The differences among these architectures lie in the ways of how the carry bits are generated and propagated. There are also several logic styles to implement the basic full-adder cell: complementary CMOS, complementary pass-transistor logic, transmission-function full adder, transmission-gate adder, 14-transistor adder, and 10-transistor adder [13]–[18]. All these different logic styles have their cost–performance tradeoffs. The selection criteria concerned include supply-voltage range, voltage swing, speed or delay, power–delay product, output skew, driving capability, and area.

All the aforementioned adder-related techniques can be used for implementing the accumulation function required in the Flying-Adder architecture. However, a recent study on the time-average-frequency and Flying-Adder architecture has revealed an interesting feature that can lead to a fast yet powerful and area-efficient implementation of the accumulation function.
for the Flying-Adder PLL [5], [6]. This paper will focus its attention on this issue.

The rest of this paper is organized as follows. Section II will discuss the newly observed feature associated with the Flying-Adder architecture and its beneficial consequence on the circuit implementation. Section III is the mathematical proof of this method. Section IV shows the behavior-level simulation. Section V is the transistor-level circuit simulation. Section VI discusses the advantage of this new scheme. Section VII presents a related interesting problem for future research. Section VIII discusses the issue of time-average-frequency and jitter.

II. ACCUMULATOR IN THE FLYING-ADDER ARCHITECTURE

A. Brief Review of Time–Average Frequency

The concept of time-average frequency has been formally introduced in [5]. The key difference between the conventionally defined frequency and time-average frequency is graphically shown in Fig. 1. As shown, for conventionally defined frequency, all the cycles must have the same length in time. However, for time–average frequency, this not necessarily has to be true. Its cycles could have different lengths. However, for a given time frame, such as 1 s, the number of cycles existing in both schemes has to be the same. The key advantage of time–average frequency is that it can make the difficult task of clock generation (frequency synthesis) less stressful.

B. Accumulator in the Flying-Adder Architecture

The Flying-Adder architecture is a frequency synthesis technique that implements the time–average-frequency concept in real circuits. Fig. 2 shows the principal idea of this architecture. As demonstrated in the numerical examples presented in [5, Section II.E], the waveform of CLKOUT is composed by utilizing the selected outputs from an N-output VCO/PLL to trigger the toggle flip-flop. Assume that the time difference between any two adjacent VCO outputs is Δ (in picoseconds) and the frequency control word is \( FREQ = I \), where \( I \) is an integer. Also, assume that the initial address value of the \( N \rightarrow 1 \) MUX is \( x \) (an integer). Then, due to the operation of accumulation, the address value will progress in the pattern of \( x, x + I, x + 2I, x + 3I, \ldots \). Each of these address values will select a corresponding output from \( N \) equally spaced VCO outputs, and the rising edge of the currently selected signal triggers the flip-flop, which, in turn, generates an edge for the output signal CLKOUT. The period (frequency) of the CLKOUT is \( 2I \Delta \).

Now, if we incorporate a fractional number into the control word as \( FREQ = I + r \) (\( r \) is a fractional number), the output CLKOUT will have a slightly different structure. When \( r \) is presented in \( FREQ \), the fractional \( r \) adds to itself repeatedly as well during the accumulation. Eventually, an overflow will happen, and a carry-in signal will propagate to the integer part. In such a case, instead of \( 2I \Delta \), the resulting waveform is \( (2I + 1)\Delta \). Therefore, the signal CLKOUT contains two types of cycles: \( 2I \Delta \) and \( (2I + 1)\Delta \). The rate of occurrence of \( (2I + 1)\Delta \) depends on the magnitude of \( r \). Overall, after a certain number of cycles, the desired time–average frequency can be achieved [5]. It is worth to mention that Fig. 2 is the principal circuit, not the working circuit used in real chips. Unlike Fig. 2 where the transfer function is \( T = 2 \times FREQ \times \Delta \), in the real circuit, the transfer function is \( T = FREQ \times \Delta \) [2]. In the following sections of simulation, we will use the real circuit.

From the previous brief description of the Flying-Adder working scheme, it is clear that the key operation is accumulation, which is realized by the VLSI circuit component adder. Fig. 3 is the conceptual circuit structure of an \((n + 1)\)-bit adder (RCA). The augend and addend are all represented on base-\( b \). The basic cell of this adder is the 1-bit full adder, with A, B, and CI as inputs and S and CO as outputs. All the 1-bit full adders are connected serially to form the adder. The CO of the previous stage is fed to the CI of the next stage. All the CIs have to be propagated to the final stage before the operation can be considered complete. The adder output is stored in a register that is controlled by a clock.

Fig. 4 is the accumulator used in the Flying-Adder architecture. As shown, its output is fed to the input for the next cycle of addition. The number stored in the register, \( A_n \rightarrow A_{n-1} \rightarrow A_{n-2} \rightarrow \ldots \rightarrow A_0 \rightarrow A_{-1} \rightarrow \ldots \rightarrow A_{-m} \), is the result for the current clock cycle. One special feature of the Flying-Adder accumulator is that its input and result are both real numbers,
with integer part $A_nA_{n-1}A_{n-2} \ldots A_0$ and fractional part $A_{-1}A_{-2} \ldots A_{-m}$. They are separated by a decimal point, as shown in the figure. In circuit operation, only the integer part is used as the address for the $N \rightarrow 1$ MUXs. The fractional part is merely used for accumulation. Only when overflow happens that it impacts the circuit’s operation. Therefore, compared to the conventional add operation where all bits are important if the correct result is desired, in Flying-Adder accumulation, only the integer part $A_nA_{n-1}A_{n-2} \ldots A_0$ and the carry-in from the fractional part convey useful information. At any given time, the exact value of the fractional part $A_{-1}A_{-2} \ldots A_{-m}$ has no impact since it is not used.

Using this observation, instead of the structure in Fig. 4, we can build a Flying-Adder accumulator using the structure of Fig. 5. The difference lies in the way the carry-ins are propagated. In this new structure, the full result of each 1-bit full adder, including the carry-in bit, is immediately fed to the registers, as shown in Fig. 5(a). The carry-in is fed to the next stage only at the next clock cycle [Fig. 5(b)]. The advantage of this scheme is recognizable: It reconfigures a large multiple-bit adder into the separated 1-bit full adders. The speed gain is unmistakable. The validity of this method will be analyzed in the next section with more details.

III. MATHEMATICAL PROOF

From the aforementioned discussion, it is understandable that we are only interested in the number of carry-ins within a given time frame. We do not care when the carry-ins actually occur and what the sequence of their occurrences. Based on these understandings, it is stated that the serially connected 1-bit full adders of Fig. 5 can be used to replace the parallel multiple-bit adder of Fig. 4. This section provides the mathematical proof.

Assume that $FREQ$ takes this value: $FREQ = I + r$, where $r$ is the fractional part $0 < r < 1$. Using base-$b$, $r$ can be represented as (1) when an $m$-bit system is employed

$$r = r_1b^{-1} + r_2b^{-2} + r_3b^{-3} + \cdots + r_mb^{-m} .$$

(1)

After $b^m$ accumulations ($b^m$ clock cycles) of using the parallel adder of Fig. 4, the sum of accumulations can be calculated as

$$S_1 = b^mr = r_1b^{m-1} + r_2b^{m-2} + r_3b^{m-3} + \cdots + r_mb.$$  

(2)

Equation (2) clearly shows that, after $b^m$ operations, all the fractional contents are propagated to the integer part. $b^m r$ is the total number of carry-ins generated during these $b^m$ operations.

Mathematically, $r$ can also be represented as

$$r = r_1b^{-1} + \cdot \cdot \cdot + r_mb^{-m} .$$  

(3)

Define

$$R_1 \equiv r_1b^{-1},$$

$$R_2 \equiv r_2b^{-2},$$

$$R_3 \equiv r_3b^{-3},$$

$$\vdots$$

$$R_m \equiv r_mb^{-m} .$$  

(4)

For each of these $R_1, R_2, R_3, \ldots$, a 1-bit full adder can be used to carry out the accumulation, as shown in Fig. 5(a). After $b^m$ clock cycles, the result becomes

$$b^m \ast R_1 = r_1b^{m-1} ,$$

$$b^m \ast R_2 = r_2b^{m-2} ,$$

$$b^m \ast R_3 = r_3b^{m-3} ,$$

$$\vdots$$

$$b^m \ast R_m = r_mb .$$  

(5)

Since the $m$ 1-bit full adders are serially connected, as shown in Fig. 5(b), the carry-ins generated at each stage are propagated forward progressively at each clock cycle. No carry-ins are lost in this process. Therefore, after $b^m$ operations, the result can be deduced as

$$S_2 = b^m R_1 + b^m R_2 + \cdots + b^m R_m = r_1b^{m-1} + r_2b^{m-2} + r_3b^{m-3} + \cdots + r_mb = S_1.$$  

(6)

Equation (6) proves that the sums of the two approaches are equal for every $b^m$ accumulations. For time–average-frequency definition and Flying-Adder implementation, this is precisely...
what is needed. Also, it is worth mentioning that only \( b \) clock cycles are needed to satisfy the \( S_0 = S_1 \) condition if \( r_x \) is the first (from LSB) nonzero bit in (1).

IV. BEHAVIOR-LEVEL SIMULATION

Matlab/Simulink has been used to verify this new accumulator idea in the behavior level. First, the accumulator itself is studied. Then, both the new accumulator and the conventional-adder-based accumulator are used in the Flying-Adder frequency synthesizer to study their impact on the clock output.

A. Study of Accumulators

A full adder, with inputs A, B, and CI and outputs S and CO, is constructed by using the logic operator of the Simulink package. A conventional 6-bit RCA is then built upon it, as shown in Fig. 6. The adder of the new approach is constructed according to Fig. 7. As shown, there are memory units between the CO and CI of adjacent full adders, which mimic the registers in the real circuit. The sum bits in all the full adders are fed back to one of the inputs (b pin) of the corresponding units, also through employing the memory units, to achieve the accumulation. However, these memory units are instantiated in the next hierarchy level (not shown here). Both adders are used to do the accumulation in the fractional part. The decimal point is immediately before the MSB. The carry-in of the full adder at the MSB position (leftmost) is the focus. Using these adders, (1) becomes (7) since base-\( b \) is two and \( m = 6 \)

\[
r = r_1 2^{-1} + r_2 2^{-2} + r_3 2^{-3} + r_4 2^{-4} + r_5 2^{-5} + r_6 2^{-6},
\]

Fig. 8 is the accumulation result when \( r = .000001b \) \((r_6 = 1\) and \( r_1 = r_2 = r_3 = r_4 = r_5 = 0) \). As expected, the addition result from the conventional adder is a linearly increased straight line since it adds one to itself at every clock cycle. After every \( 6^m = 2^6 = 64 \) cycles, a carry-in is generated at the MSB position. For the case of the new approach, the result is different. Most of the time, the result at each clock cycle is not correct since it is different from that of the linear curve. In addition, the locations of the carry-ins of the two approaches are different as well. However, after every 64 cycles, there is one carry-in generated for both cases, just as proved in Section III.

Fig. 9 is the case of \( r = .001001b \) \((r_3 = r_6 = 1\) and \( r_1 = r_2 = r_4 = r_5 = 0) \). As expected, the accumulation result increases monotonically for the conventional case. Furthermore, for the new approach, the result is different. However, for any window of 64 cycles, the numbers of carry-ins generated are
the same for both approaches, which are nine in this setting. Fig. 10 shows the total number of carry-ins after a long time of simulation (700 clock cycles). The first and fourth rows (from the top) are the adder outputs at each clock cycle. The third and sixth rows are the carry-ins generated. The second and fifth rows are the total numbers of carry-ins accumulated at the current time. For the Flying-Adder operation, the information presented in the second and fifth rows is the careabouts. It is clear that both approaches produce the same output.

B. Testing the Accumulators Within the Frequency Synthesizer

The two types of accumulator (Figs. 6 and 7) have both been used in the Flying-Adder synthesizer Simulink model to compare the results. As discussed in [5], the frequency control word \( FREQ(= I + r) \) determines the average frequency. The fraction \( r \) defines the shape of the output clock’s spectrum. Therefore, during the comparison, we will study the following: 1) the numerical value of the average frequency and 2) the shape of the spectrum.

The VCO used in the simulation is running at 1 GHz (1-ns period) and has 32 equally spaced outputs. The time difference \( \Delta \) between any two adjacent outputs is \( 1 \text{ ns} / 32 = 31.25 \text{ ps} \). Fig. 11 is the case of \( FREQ = 62.125 \). Under this control word, the output frequency should be 1.9414 ns (515 MHz) theoretically. The simulation results, from the two structures of using the conventional accumulator and the new one, are both 1.9413 ns. The simulation time covers 6181 clock cycles. Fig. 11 shows the clock pulses and the trend of the average frequency (period) of the first 130 cycles. The clock pulses are simply added to the frequency plot, although it is not in the same unit as period (\( Y \)-axis), to save space.

Fig. 12 shows the outputs’ spectra calculated from 6181 cycles of simulations. The \( X \)-axis is the discrete Fourier transform point which corresponds to the real frequency. It shows that the average frequencies (the main stems) of both cases are at the same location. Also, the numbers of spurs and their locations are the same as well. This indicates the fact that both accumulators have achieved the same function in the Flying-Adder synthesizer operation.

Fig. 13 is another example where \( FREQ = 62.015625 \). Theoretically, the average frequency should be 1.9380 ns (516 MHz). The simulations of both approaches result in 1.9378 ns after a simulation time of 6192 clock cycles. These plots also show that the main stems of both cases are at the same location, and so are the spurs.

V. TRANSISTOR-LEVEL CIRCUIT SIMULATION

Real circuits using both types of accumulator have been built in a 90-nm CMOS process. The input reference is a 27-MHz crystal. The VCO is running at 1.08 GHz (40 \( \pm \) 27 MHz, 925.925 ps). There are eight equally spaced outputs from the VCO. Thus, the time difference between any
two adjacent VCO outputs, $\Delta$, is $925.925/8 = 115.74$ ps. Fig. 14 shows the simulation result of $FREQ = 12$ and 12.0625, using the accumulator of the new approach. For the case of $FREQ = 12$, the calculated frequency should be $T = FREQ \times \Delta = 12 \times 115.74$ ps = 1.389 ns (720 MHz). For $FREQ = 12.0625$, the calculated frequency is 1.396 ns or 716.27 MHz. As shown in the plot, the simulation results are 720.01 MHz for $FREQ = 12$ and 716.27 MHz for $FREQ = 12.0625$. The spurs associated with $FREQ = 12.0625$ is the modulation caused by the 0.0625 fraction. From [5, eq. (8)], the space between the spurs can be roughly calculated as $0.0625 \times 716.27 = 44.7$ MHz. This agrees with the simulation, as shown in the figure.

Fig. 15 shows the simulation cases for the circuit of using the conventional accumulator. The frequency control word settings are the same as in the case of Fig. 14. Figs. 14 and 15 both demonstrate the following: 1) Both types of accumulator behave similarly when used in the Flying-Adder synthesizer, and 2) the results agree with what is expected from the calculation. Several other cases of $FREQ$ settings have been simulated using these circuits as well; the results also support the aforementioned 1) and 2).

Fig. 16 is the simulation result of Flying-Adder spread-spectrum clock generation. The $FREQ$ setting is 12.0625. A triangular-wave modulation pattern is applied at $FREQ$. The modulation depth is about 1%. The modulation method is down spread. The circuit used is the one with a new accumulator. This simulation further proves that the new accumulator is working correctly for the Flying-Adder synthesizer.

As another test, the new accumulator is used in the simulations of switching-speed test. One of the key advantages of the fixed-VCO-Flying-Adder architecture is its instantaneous response speed. Fig. 17 is the simulation which shows this feature. In this example, at 5 $\mu$s time, the frequency control word $FREQ$ is changed from $FREQ = 10$ to $FREQ = 11$. SPICE simulation shows the instant frequency switching from 864 to 785.46 MHz. The second row is the output clock waveform, which contains a train of clock pulses (the detailed waveform can only be visible by zooming in closely).

Fig. 18 shows a more complicated example that involves the fractional part. At 5 $\mu$s time, the control word is changed from 10 to 10.0625. Due to the 0.0625 fraction, there is one carry-in generated in every 16 cycles. This causes the instant frequency (period) changed from normal 10$\Delta$ to longer 11$\Delta$ at the rate.
Fig. 17. FREQ switches from 10 to 11 at 5-μs time.

Fig. 18. FREQ switches from 10 to 10.0625 at 5-μs time.

of once every 16 cycles. This can be clearly seen from the frequency measurement in the second row (from the top). The first row shows the time–average frequency measured at every 16 cycles. Figs. 17 and 18 serve as another testimony of the validity of the new accumulator.

VI. ADVANTAGES OF THE NEW SCHEME

In this section, we will use the real circuits designed in a 90-nm CMOS process to compare the new accumulator and the conventional accumulator in several aspects, including circuit speed, silicon area, power consumption, and design effort.

A. Speed Comparison

Fig. 4 is the accumulator in conventional fashion. Clearly, the circuit speed is closely tied to the size or width of the adder. On the other hand, if the fractional accumulator is constructed using the structure of Fig. 5, the speed of this fractional accumulator will be equivalent to that of a 1-bit full adder. Consequently, using this new scheme, the accumulator speed will only depend on the width of the adder in the integer part, which is usually much smaller. For the VCO of 32 outputs, the adder size is 5 bits. It will be 4 or 3 bits for the VCOs of 16 or 8 outputs, respectively. Table I lists the logic synthesis results of the circuit speeds of various adder sizes, using the 90-nm process. The integer part is 3 bits. As expected, the circuit speeds vary with the circuit sizes. For the case of the new scheme, the speed is fixed at 0.43 ns. This is due to the fact that the integer part is always 3 bits, regardless of the whole adder size.

B. Area Comparison

Table II is the area comparison. The base unit for calculation is the 1 drive two-input NAND gate from the library. As shown, the area increases with width. The first number inside each parenthesis is the size of combinational logics; the second number is for sequential logics (registers). Comparing the two schemes, the conventional accumulator definitely uses more area for all the four cases. It is interesting to see that the new accumulator uses more sequential area. This is due to the fact that it has two registers for every 1-bit full adder [Fig. 5(a)]. While in the conventional one, only one register is needed for each bit. However, the table does not show a 2:1 ratio because the exact register usage ratio should be \((2^w - 3)/2^w\), where \(w\) is the width of the whole adder. Also, several different types of register are used in these adders, and those different registers have different physical sizes.

C. Power-Consumption Comparison

Table III lists the power-consumption numbers used by the accumulators when they operate at speeds of 1 GHz, 500 MHz, and 100 MHz. The same input stimulus is used for both types of accumulator. The stimulus is constructed in such a way that the activity is close to 100%. In other words, the outputs at each
bit are toggling at their maximum rates. In a real working environment, this hardly happens, and consequently, the power-consumption number will be lower. The supply voltage is 1.1 V. This table shows that the new accumulator uses roughly one-half of the power consumed by the conventional one for all three operating frequencies. Also evident is that power consumption is linearly proportional to operating speed.

D. Design-Effort Comparison

The data collected in Tables I–III demonstrate the advantages of the new scheme in terms of area, speed, and power consumption. Aside from these benefits, another interesting comparison is the design effort. For the case of the conventional accumulator, the design has to be carried out in the fashion of HDL coding, logic simulation, and logic synthesis. This requires an HDL simulation tool and a synthesis tool (and associated license fees). Moreover, the designer is required to have the HDL coding skill and the knowledge of using the aforementioned tools. This is often not the case for analog-circuit designers. On the other hand, using the new scheme, the designer can easily build the accumulator by a schematic capture tool, which analog designers use every day.

VII. Problem for Future Work

As discussed in [5] and shown in Figs. 12 and 13, there are spurious components in the output clock’s spectrum when a fractional number is used. These spurs can be converted into noise by adding a modulation signal to $FREQ$, as shown in [5, Fig. 23]. The modulation function could be a random number generator. For example, for the case of $FREQ = 62.125$ (Fig. 12), the spurs can be cleared out, as shown in the right plot of Fig. 19. As another example, Fig. 20 shows the result of $FREQ = 62.015625$ (Fig. 13). These plots demonstrate that random wandering can break the periodical characteristic associated with a fractional number and consequently eliminate the spurs. This brings up an interesting question: Can the random mechanism be directly built into the accumulator to save the silicon area of extra circuitry?

Referring back to Figs. 8 and 9, it is clear that the occurrences of carry-ins are totally different between the two types of accumulator. The carry-ins generated by the conventional accumulator is periodical and completely predictable, while the carry-ins produced by the new scheme is unpredictable (i.e., the mechanism has not been understood yet). However, the pattern still seems periodic, which produces spurious components. Thus, the question is:

Is it possible to build an accumulator which can achieve the same average effect (i.e., generate the same number of carry-ins in a given time frame) but break the periodical characteristic?

VIII. Time-Average-Frequency and Jitter

Clock signal is one of the most important signals in any VLSI chip. When clock signal is used in systems, other than frequency, the most crucial characteristic is its edge uncertainty. The source of clock edge uncertainty can be investigated from two directions: the uncertainty caused by the generator (PLL, crystal oscillator, etc.) and the uncertainty caused by the clock distribution network (clock tree). The uncertainty of the latter type is also termed as clock skew. The issue of clock skew involves more than one clock sink, often hundreds or thousands of sinks. This type of edge uncertainty will not be discussed in this paper since it has already been intensively studied [19]–[21].

The edge uncertainty associated with the clock generator can be caused by the phase noise induced by white noise, flicker noise, and thermal noise [22]–[27] or by modulations activated by certain deterministic signals, such as the input reference and the noise in power supplies. This type of edge uncertainty is usually called clock jitter. Time-average frequency, by its definition, introduces clock edge movement [5]. However, this edge movement is NOT uncertain but precisely controlled. Therefore, it is fundamentally different than the jitter and skew.

As addressed before, when the control word takes $FREQ = I + r, 0 < r < 1$, the resulting clock waveform is composed of two types of cycles: $I \Delta$ and $(I + 1) \Delta$. The rate of $(I + 1) \Delta$ cycle occurrence depends on $r$. For the case of $FREQ = 12.0625$ (Figs. 14 and 15), there are 15 cycles of $12 \Delta$ and one cycle of $13 \Delta$ for every 16 cycles. Fig. 21 is the histogram of instantaneous periods from SPICE simulation. The instantaneous periods are distributed as two separate branches, i.e., left of $12 \Delta$ and right of $13 \Delta$. The $r = 0.0625$ fraction results in 6.25% of the total cycles on the right and 93.75% of the total cycles on the left. The average frequency is somewhere in between the left and
right branches. The mission of time–average frequency can be appreciated from the following calculation:

\[ 16 \times 12.0625 = 15 \times 12 \Delta + 1 \times 13 \Delta \]

In other words, if we have difficulty in creating a PLL that shall generate a frequency of 716.33 MHz, we can use an FAPLL of \( FREQ = 12.0625 \) to mimic this frequency. From the application point of view, the end result is the same: Within 22.34 ns, there are 16 operations. Figs. 22 and 23 are two additional examples to illustrate this point. Using the evidence of Figs. 14, 15, 18, and many others, we can also opine that fractional number is the “controlled noise,” which can result in the desired frequency shift precisely. This statement is more persuasive when \( r \) is small, such as in the case of DCXO [8]. A fractional-\( N \) PLL also uses a fractional number to achieve some average effect. However, the resulting clock signal cannot be qualified as time–average-frequency. Its clock edge distribution is not in the shape of these shown in Figs. 21–23. Its edge movement is not directly controlled.

When the time–average-frequency-based clock signal is used to drive a digital system, the timing closure (setup check) should be carried out by using the constraint of the left branch (12\( \Delta \) in Figs. 21 and 22 and 11\( \Delta \) in Fig. 23). As long as the system can be timing closed, the size of \( \Delta \) should not be included in the jitter calculation. Jitter is the edge uncertainty of the left branch (Gaussian distribution). Time–average-frequency does not have any impact on hold check since hold check only concerns one clock edge.

When the time–average-frequency-based clock signal is used to drive systems that have embedded analog components (ADC/DAC), the spurious signals introduced by fractional number certainly need attention. In the case of fractional-\( N \) PLL, the spurs are usually close to the carrier since the carrier is modulated by the frequency of fractional number times the input reference frequency. These close-in spurs usually have harmful impacts on system operation. However, in the Flying-Adder PLL case, the spurs are far away from the carrier. The reason is that the spurs are calculated from fractional number times the carrier frequency. This is evident in Figs. 14 and 15 where the spurs are \(~44\) MHz away. When fractional number is extremely small, the spurs do move closer to the carrier. However, the magnitude of the spurs in these cases decreases significantly (Fig. 20 and [5]). Therefore, the spurious signal in FAPLL is not as severe as in the case of fractional-\( N \) PLL.

The time–average-frequency-based clock signal has been used in commercial products for almost ten years. Not a single case of problem, related to time–average-frequency, has been reported. Fig. 24 is the case where the time–average-frequency-based clock signal is used to drive the LVDS transmitter. In Fig. 24(a), an FAPLL is used to generate 85.5 MHz for graphic WXGA mode (1360 × 768 at 60 Hz). This clock is used to drive a 7\( \times \) PLL to serialize the data. The
top waveform is the data, the bottom is the clock. The setting of FAPLL is as follows: \( f_r = 27 \text{ MHz} \), \( P = 1 \), \( N = 38 \), \( K = 8 \), \( FREQ = 12 \), and \( M = 8 \). This results in \( f_0 = 85.5 \text{ MHz} \). \( f_r \) is the prescaler divider ratio, \( N \) is the PLL divider ratio, \( K \) is the number of VCO outputs, \( M \) is the postdivider ratio [please see [7, Fig. 2 and eq. (4)]]. In this case, there is no fractional number used in \( FREQ \) (i.e., it is not time–average-frequency). In Fig. 24(b), the FAPLL setting is \( f_r = 27 \text{ MHz} \), \( P = 1 \), \( N = 45 \), \( K = 8 \), \( FREQ = 11.36842 \), and \( M = 10 \). This also results in \( f_0 = 85.5 \text{ MHz} \). \( \Delta \) is about 103 ps in this case. The peak-to-peak jitter in case (a) is about \(~64\) ps. It is about \(~70\) ps in case (b) (left branch). There is hardly any visually noticeable difference between (a) and (b).

**IX. CONCLUSION**

This paper has demonstrated a new scheme for constructing the accumulator used in the Flying-Adder frequency synthesis architecture. The validity of the new scheme has been proved from several directions. The advantages of this new scheme are faster circuit speed, smaller circuit area, and lower power consumption. This scheme solves a long-lasting problem in the implementation of the Flying-Adder circuitry and will have profound impacts on all future Flying-Adder applications. We believe that this new approach can also be used in the accumulator of fractional-\( N \) PLL since the carry-in is the only information needed in that case as well. The important issue of time–average frequency and jitter has also been addressed in this paper.

**REFERENCES**


Lining Xiu (M’95–SM’03) was a Senior Member of the Technical Staff of Texas Instruments Incorporated, Dallas, TX. He has done significant amount of work/research in the area of frequency synthesis. He is the Principal Inventor of the Flying-Adder frequency and phase synthesis architecture, which has been used in many commercial products. He is also an expert in VLSI system-on-a-chip integration, with battle-proven integration experience on several very large chips. In this area, he has one book entitled “VLSI Circuit Design Methodology Demystified: A Conceptual Taxonomy” (Wiley, 2007). He is currently the IEEE Circuits and Systems Society Vice President of Regions 1–7 for 2009–2010.