electronics

Article

A Fast Lock-In Time, Capacitive FIR-Filter-Based Clock Multiplier with Input Clock Jitter Reduction

Zhaoquan Zeng [1,2], Ling Zhang [1,2], Lijiao Gong [1,2,*] and Ning Zhang [1,2]

[1] School of Mechanical and Electrical Engineering, Shihezi University, Shihezi 832000, China
[2] Xinjiang Production and Construction Corps Key Laboratory of Advanced Energy Storage Materials and Technology, Shihezi University, Shihezi 832000, China
* Correspondence: glj_mac@shzu.edu.cn

Abstract: This paper presents a fast lock-in time clock frequency multiplier without using traditional clock generation circuits such as PLLs and DLLs. We propose a novel technique based on capacitive finite impulse response (FIR) filters to generate clock phases while reducing the input clock phase noise at the same time. A new delay line circuit is also proposed for improving power supply rejection. In addition, to improve the matching quality as well as the end-effects tolerance of the on-chip capacitors, a single-value series/parallel algorithm is proposed. Designed in a 0.18 μm digital CMOS process, with a 20 MHz input clock frequency, the multiplier achieves a multiplication factor of 5 with a lock-in time of less than 4 clock cycles. The input clock jitter is reduced from 7ns RMS to 153 ps RMS after frequency multiplication.

Keywords: frequency multiplier; electronics design automation; phase-locked loop; delay-locked loop; capacitor network

Citation: Zeng, Z.; Zhang, L.; Gong, L.; Zhang, N. A Fast Lock-In Time, Capacitive FIR-Based Clock Multiplier with Input Clock Jitter Reduction. *Electronics* **2023**, *12*, 1439. https://doi.org/10.3390/electronics12061439

Academic Editors: Teo Tee Hui, Akram Ben Ahmed and Costas Psychalinos

Received: 7 January 2023
Revised: 14 March 2023
Accepted: 16 March 2023
Published: 17 March 2023

1. Introduction

The clock frequency multiplier has many applications in integrated circuits, especially for modern system-on-chip (SoC) designs [1]. In general, there are a few methods to realize frequency multiplication: phase-locked loops (PLLs) [2–4], delay-locked loops (DLLs) [5–8], and clock phase interpolation [9–12]. PLLs and DLLs offer good solutions for accurate clock generation; however, they generally require a long time to lock or settle due to the feedback operation. In addition, DLL/PLL-based circuits require substantial amounts of design effort and time, and experienced designers are needed to migrate the same functions from one process to another [13]. On the contrary, clock phase interpolation methods offer a solution for producing a multiplied frequency with significantly reduced lock/settle time, less power consumption, and smaller silicon area [14]. These methods, therefore, considerably reduce the overall cost of the design and accelerate time-to-market for new designs [15]. Since the clock multipliers in this category are generally digital intensive, it is very convenient to make them portable among different processes. Several clock interpolation-based frequency multipliers have been proposed. Saeki et al. [11] uses the divider to generate the primary phase and direct clock cycle interpolation to generate $2N$ times the input frequency. Yin et al. [12] adopt the passive RC polyphase filter (PPF) to generate the primary phases that are then interpolated to obtain the necessary sub-phases. However, as it is well known, using clock dividers or PPF to generate primary phases causes large phase errors due to device or layout mismatches, which result in degraded jitter performance.

In order to overcome the long locking time, high design effort, as well as high power and silicon budget of PLL and DLL clock multipliers, it is necessary to investigate the viability of designing clock multipliers using novel clock phase interpolation techniques. This motivation leads us to the research of designing a clock frequency multiplier based on finite impulse response (FIR) filters. As it is shown in Figure 1, the proposed FIR-filter-based clock multiplier has 4 stages. The capacitive primary phase generator (CPPG) is the first stage of the clock multiplier and is composed of a tunable delay line and a capacitive network that embodies the FIR filter coefficients. It is employed to generate highly accurate differential primary phases and reduces the input clock jitter concurrently due to its inherent filtering property. Based on the primary set of phases generated by the CPPG, a capacitive sub-phase generator (CSG) is used to generate a set of arbitrary differential sub-phases that are to be followed by a zero-crossing detector (ZCD). The edge combiner (EC) combines all the M-phase clock signals to generate a signal at M times of the input frequency f_{in}. It is worth mentioning that such a clock multiplying technique is also enabled by the proposed single-value series/parallel algorithm for the capacitive networks used in both CPPG and CSG blocks. The algorithm effectively improves the matching quality and end-effects tolerance of the on-chip capacitors, making the designed coefficients accurate and resilient to process variations.

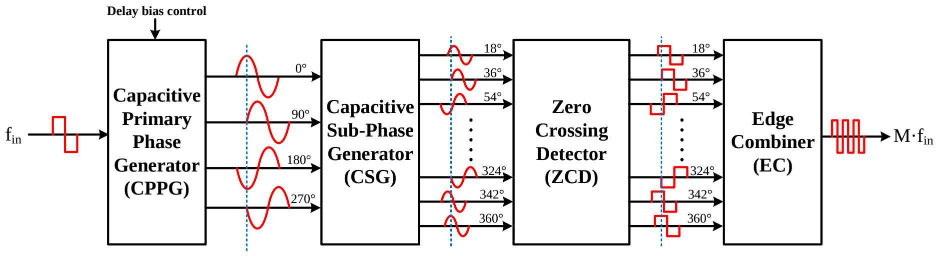


Figure 1. Architecture of FIR-filter-based clock multiplier.

2. Principle of Operation and Circuit Implementation

2.1. Capacitive Primary Phase Generator

2.1.1. Fundamental Mathematics

Synchronous digital filter design techniques laid the foundation of CPPG design. Several classic and recently-published digital filter design books and papers deliver a detailed tutorial on the FIR filter design topic, which provide strong mathematical support to the CPPG design [16–21]. It can be proved that if a sinusoidal signal $\sin(\omega t)$ convolves with another sinusoidal signal with certain phase shift $\sin(\omega t + \theta)$, the output signal after the convolution will carry the phase shift of θ. Such a feature can be exploited to implement multiphase FIR filters whose impulse response has controllable phase information. For example, two sub-FIR filters implemented with $h_1[n] = \sum_{k=0}^{N} \sin(\omega \cdot \tau \cdot n)\delta[n-k]$ and $h_2[n] = \sum_{k=0}^{N} \sin(\omega \cdot \tau \cdot n + \pi/2)\delta[n-k]$, where ω is the angular frequency, τ is a unit delay, and N is the number of the FIR taps, will have the same magnitude response but $\pi/2$ of phase difference. In other words, by feeding the two sub-FIR filters with the same input signal, the two output signals with the same amplitudes but exact $\pi/2$ phase difference are generated. Such properties are perfect to be employed for precise primary phase generation.

In order to build the multiphase FIR filters for discrete systems and suppress the occurrence of Gibbs phenomena due to truncation, a window function is added:

$$h_i[n] = \sum_{k=0}^{N} K_{\alpha=3} \sin(\omega \cdot \tau \cdot n + \theta_i)\delta[n-k], \tag{1}$$

where $K_{\alpha=3}$ is the Kaiser window with parameter $\alpha = 3$. Mathematically, the FIR filters with varying θ_i have the same magnitude response and constant phase difference in phase response at the frequency of interest (i.e., input clock frequency). As an example, a set of two sub-FIR filters with a central frequency of 20 MHz and relative phase is built with the unit delay and tap number set as 1.5 ns and 80, respectively. The impulse responses of the two sub-FIR filters are shown in Figure 2. As can be seen, the filters have the same magnitude response but also have linear phase responses with a constant phase difference of 90° between each other. Meanwhile, high out-of-band signal suppression provided by the FIR filters helps to reduce the input clock jitter.

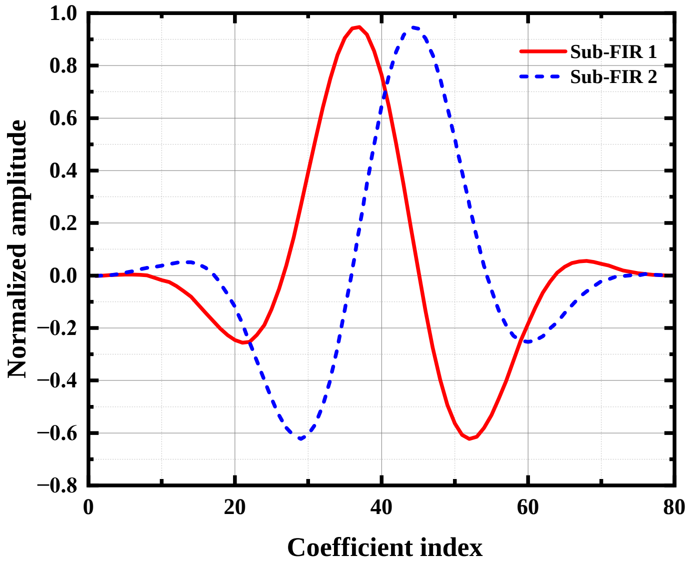


Figure 2. Impulse response of two sub-FIR filters.

It is worth mentioning that a multiple-frequency or wideband multiphase FIR filter can be constructed in a similar way as described above. The accuracy of the output phases generated by the multiphase FIR filters is mainly determined by the unit delay and number of taps in the delay line. In general, the unit delay and the number of taps can be determined in such a way that their product is comparable to a few periods of the input clock (typically, 2–3 periods). However, when the number of taps (or the length of the delay line) is fixed, a larger unit delay may degrade the phase accuracy. This is mainly because a larger unit delay has a coarser resolution for the FIR filters in the time domain, and it degrades the phase accuracy during reconstruction. It can be confirmed that 100 ps of the RMS unit delay error only leads to about 0.15 degree of phase error in simulation, which is about 15 ps peak-to-peak jitter for a 20 MHz clock frequency. Compared to the DLL counterpart, the fact that the output phase accuracy is marginally dependent on the unit delay accuracy in the proposed technique helps to substantially reduce the design efforts.

2.1.2. Circuit Implementation

The multiphase FIR filters indicated in Equation (1) can be demonstrated as signal flow chart in Figure 3, where K_1, K_2, ... K_{16} are the coefficients of FIR filters, U_1, U_2, ... U_8 represent the unit delay elements. The FIR filters shown in Figure 3 can be then constructed from a Thevenin equivalent network as shown in Figure 4 where we take a set of two FIR sub-filters with phase $\theta_1 = 0°$ and $\theta_2 = 90°$ as an example. The FIR filters are implemented by a star connection from all of the signal sources through a capacitor to eliminate the impact of resistive thermal noise. The values of capacitors C_1, C_2, ... C_8 are corresponding with the filter coefficients K_1, K_2, ... K_8, which are used to generate signals with phase $\theta_1 = 0°$. Similarly, capacitors values C_9, C_{10}, ... C_{16} correspond with the filter coefficients K_9, K_{10}, ... K_{16}, which are used to generate signals with phase $\theta_2 = 90°$. By doing Thevenin analysis on the circuits shown in Figure 4, we can derive the output of the FIR filters as

$$V_{out} = \frac{1}{SC_p} \cdot \sum_{n=0}^{N} \frac{V_n}{\frac{1}{SC_n}}, \tag{2}$$

where V_n are the signal sources, C_n are the star-connected capacitors, and C_p is the parallel value of all the capacitors. The values of the capacitors C_n can be conveniently calculated by combining (1) and (2) if τ and θ are given.

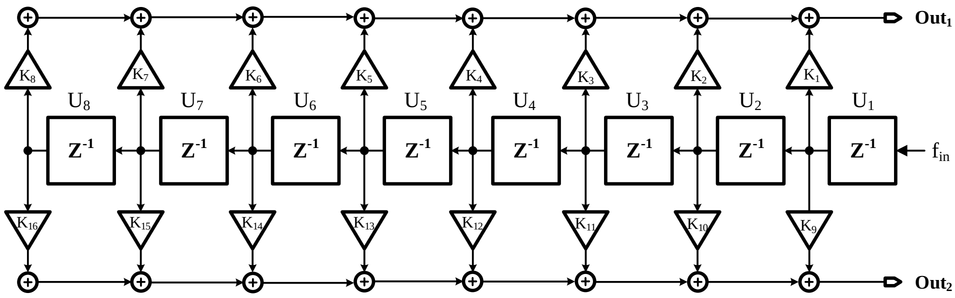


Figure 3. Structure of the capacitive multi-phase FIR filter with two phases output.

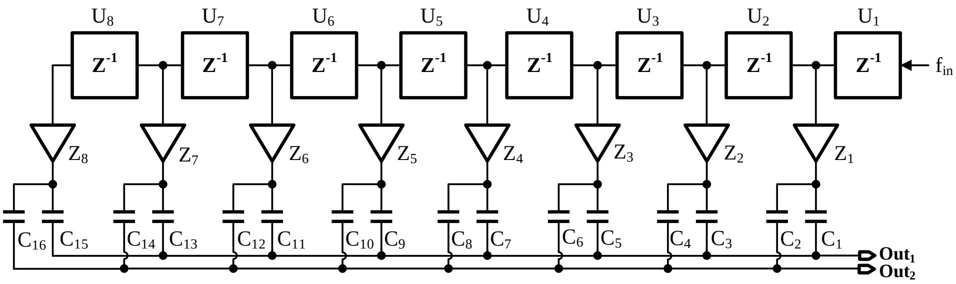


Figure 4. Capacitive multi-phase FIR filter with two sub-filters.

In Figure 4, the unit delay indicated in Equation (1) is implemented by the delay elements U_1, U_2, … U_8, which is followed by the buffers Z_1, Z_2, … Z_8 driving two sets of capacitors concurrently. The capacitors connected to Out_1 form the sub-filter with phase shift $\theta_1 = 0°$ and those connected to Out_2 form the sub-filter with phase shift $\theta_2 = 90°$. In order to cope with negative values in the coefficients, a differential output scheme can be used. That is, the output signal Out is separated into a pair of outputs: Out^+ and Out^- and, for example, the outputs of $\theta_1 = 0°$ can be expressed as $Out_1 = Out_1^+ - Out_1^-$, where Out_1^+ is connected when the corresponding tap is positive, otherwise Out_1^- is connected. Although such a scheme is able to implement negative coefficients, the number of taps connected to Out_1^+ and Out_1^- can be different, resulting in imbalanced output impedance between Out_1^+ and Out_1^-. In this paper, we propose to use an inverting delay line where every consecutive output is inverted. Such design assures that the impedance at Out_1^+ and Out_1^- are approximately the same and no systematic error is produced. In addition, an inverting delay line mitigates the accumulation of rise–fall time mismatch when the input clock is propagating in the delay line, making the delay line uniformly spaced. Since the output signals after the FIR filters contain aliasing frequency contents that are associated with the unit delay, reconstruction filters are required to construct smooth analog signals for the next stage. Since the aliasing frequencies are much higher than the desired signals, the reconstruction filters can be built by simply adding certain capacitance at the output of the capacitive network. As aforementioned, the unit delay accuracy is not critical in the proposed system, but the power supply variations can have an impact on the unit delay time. Thus, in this paper, we propose a delay-line circuit that is independent of supply voltage to improve the overall system robustness.

Figure 5 shows the delay line circuit comprising three unit delay elements U_1, U_2, U_3 connected together. The output node PMOSOUT of each unit delay element is connected to the gate of the PMOS transistor in the next unit delay element, while the output node NMOSOUT of each unit delay element is connected to the gate of the NMOS transistor in the next unit delay element. The gate of each PMOS bias transistors M_3, M_6, and M_{10} are connected to a single bias voltage V_{Pbias}, and the gate of each NMOS bias transistors M_3, M_6, and M_{10} are similarly connected to a single bias voltage V_{Nbias}.

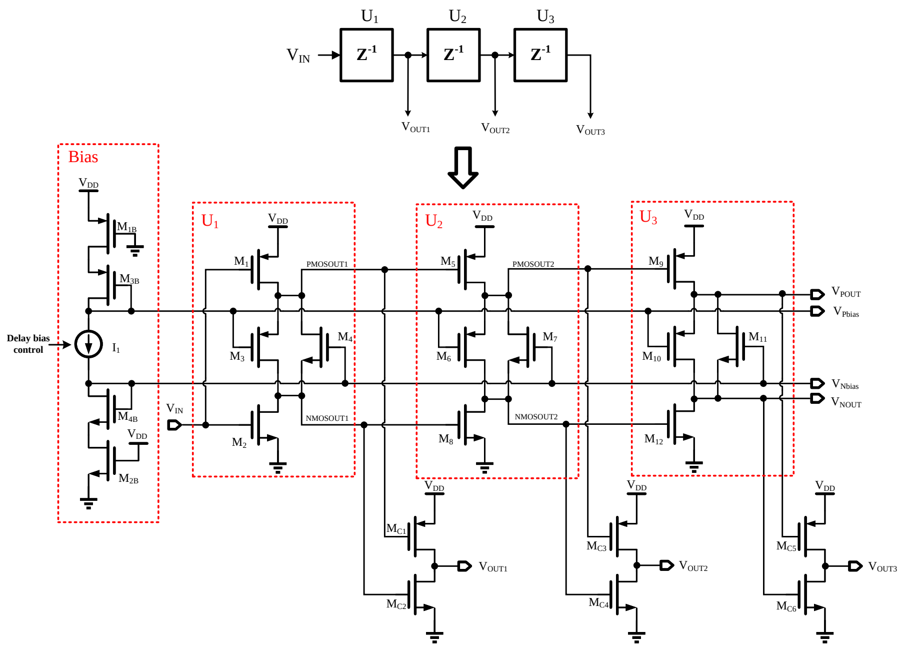


Figure 5. Delay-line circuit independent of supply voltage.

To see how the bias transistors control the delay of the delay line, we assume a high input signal at the input node V_{IN}. When the input V_{IN} is high, transistor M_2 will turn on, and the value of NMOSOUT1 from transistor M_2 will be pulled to ground with no delay. This will, in turn, cause the next NMOS transistor M_8 to immediately turn off, as there is no propagation delay from the now-low output of transistor M_2 to the gate of transistor M_8. Since transistor M_8 is now off, no signal can propagate from M_2 to M_{12}. On the other hand, the only way for NMOS transistor M_{12} to turn on is to receive the high voltage V_{DD} through transistors M_6 and M_7, which in turn only receive that voltage when PMOS transistor M_5 is on. Further, PMOS transistor M_5 only receives a low input signal from transistor M_2 through transistors M_3 and M_4. Thus, any signal reaching transistor M_{12} must pass through transistors M_3 and M_4 first, and then through M_6 and M_7.

When the signal at input node V_{IN} changes, the new signal will be propagated down the delay line at a speed dictated by the delay of each unit delay element as limited by the bias transistors rather than by the speed of the transistors that accept the input signal and provide delayed output signals. If the bias lines V_{Nbias} and V_{Pbias} are provided with voltages derived from a constant current, the delay will be constant and independent of the power supply voltage V_{DD}. Current Source I_1 provides a tunable constant current independent of the power supply voltage, and transistors M_{3B} and M_{4B} define the PMOS bias voltages. By tuning the current source I_1, the unit delay time can be adjusted to keep track of different input frequencies. A digital-controlled multi-bit current tuning mechanism can also be applied in this scenario to achieve fine-tuning of the bias current of the delay line. PMOS transistor M_{3B} is selected to have the same transconductance as bias transistors M_3, M_6, and M_{10}. While NMOS transistor M_{4B} is similarly selected to have the same transconductance as NMOS bias transistors M_4, M_7, and M_{11}. M_{3B}, and M_{4B} will define the Voltages on the bias lines as constant to first order and cause the delay line to have a nearly constant delay. Since M_{1B} is selected to have the same transconductance as M_1, M_5, and M_9, M_{2B} is selected to have the same transconductance as M_2, M_8, and M_{12}. The variation in degree to which transistors M_{1B} and M_{2B} turn on due to variations in the power supply voltage V_{DD} is the same as the variations in transistors M_1, M_5, and M_9, and M_2, M_8, and M_{12} respectively. Thus, the addition of transistors M_{1B} and M_{2B} provides compensation for variations in the power supply voltage V_{DD} and results in a more constant delay time. In a typical case, a change in the delay time of a delay line due to changes in the power supply voltage might be as great as +30%, while the use of a circuit such as the circuit of Figure 5 can reduce the change in the delay time to less than l%, making the delay of the input signal essentially independent of the power supply voltage V_{DD}. Inverter pairs M_{C1} and M_{C2}, M_{C3}, and M_{C4}, M_{C5}, and M_{C6}, are used to invert the delayed signal V_{OUT1}, V_{OUT2}, and V_{OUT3} for each tap respectively.

2.2. Capacitive Sub-Phase Generator

Arbitrary sub-phases can be generated using a 5-capacitor network as shown in Figure 6a, if given certain primary phases. P_1 and P_{1b}, P_2 and P_{2b} are the two differential inputs of the CSG, and P_O and P_{Ob} are the differential output of the CSG. We assume P_1 has an input signal $A\sin(\omega t+\theta)$, and P_2 has an input signal $A\sin(\omega t+\theta+\frac{\pi}{2})$. By analysing half of the CSG network in Figure 6b, we can get the output of CSG as

$$P_O = A\cdot\frac{1}{SC_5}\sqrt{\frac{1}{\left(\frac{1}{SC_5}+\frac{1}{SC_1}\right)^2}+\frac{1}{\left(\frac{1}{SC_5}+\frac{1}{SC_2}\right)^2}}\cdot\sin\left[\omega t+\theta+\arctan\left(\frac{\frac{1}{SC_3}+\frac{1}{SC_1}}{\frac{1}{SC_3}+\frac{1}{SC_2}}\right)\right]. \tag{3}$$

After simplification, we have

$$P_O = \frac{A\cdot\sqrt{\left({C_1}^2(C_2+C_5)^2+{C_2}^2(C_1+C_5)^2\right)}}{(C_1+C_5)(C_1+C_5)}\sin\left[\omega t+\theta+\arctan\left(\frac{C_2C_1+C_2C_3}{C_1C_2+C_1C_3}\right)\right]. \tag{4}$$

We notice in Equation (4) that the phase and amplitude of output signal P_O are independent of the signal frequency, and they can be determined by choosing the capacitor values properly. As it is shown in Figure 7, the phasor diagram where the differential primary phases P_1 and P_1', P_2 and P_2' on the circle of primary phases (P circle) are used to generate the differential sub-phase S_1 and S_1' on the sub-phase circle (S circle). Compared to the commonly used active phase interpolator, the capacitive network has no thermal noise and is linear. The generated sub-phases are much more resilient to the process, voltage and temperature (PVT) variations.

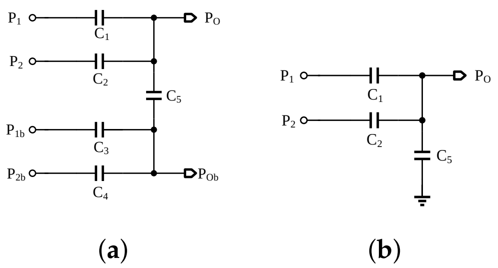


Figure 6. Circuit architecture of capacitive sub-phases generator. (**a**) Fully differential capacitive network for sub-phase signal generation. (**b**) Half capacitive network for sub-phase signal generation.

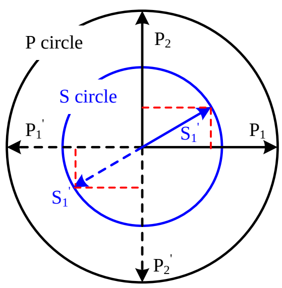


Figure 7. Phasor diagram of capacitive sub-phases generator.

2.3. Single-Value Series/Parallel Algorithm

Since capacitors are extensively used in both CPPG and CSG, the capacitance ratio accuracy and matching of the capacitors are of paramount importance to the output phase accuracy. For example, the capacitors used in the CPPG are expected to vary in a wide range: the large coefficients in the impulse response of Equation (1) are translated into small capacitance (e.g., 20 fF), while the small coefficients are translated into large capacitance (e.g., 250 fF). It is very difficult, if not impossible, to use the traditional layout techniques to build such capacitors with arbitrary capacitance in a symmetrical and matched configuration. We propose a method that represents each individual capacitor by combining a group of single-value capacitors that are in series, parallel, or both. For instance, given a set of capacitors with arbitrary values of 100 fF, 115 fF, 376 fF, 567 fF, and 1000 fF, we can use one single-valued capacitor of 280 fF to build all these capacitors with no mathematical errors. One of the possible connections is shown in Table 1, where the operators "+" and "$\|$" indicate the series connection and parallel connection, respectively. For instance, $C_u \| C_u \| C_u \| (C_u + C_u)$ represents that a total of 5 unit capacitors are used in a configuration, three parallel capacitors are in a chain with two capacitors in series.

Figure 8 shows the flowchart of the proposed recursive single-value series/parallel algorithm, where $C_{Nominal}$ is the value of the unit capacitor, C_{Target} is defined as desired value of the compound capacitive element, and C_{Result} is the capacitance of the compound element at each step in the process of the algorithm. In step 1, the variable C_{Target} is defined and given a desired value. Additionally, in step 1, the variable C_{Result} is also initiated as zero and a null set of elements. At step 2, C_{Target} is compared to $C_{Nominal}$. If C_{Target} is greater than $C_{Nominal}$, then one or more nominal capacitors should be added in parallel to get a value greater than $C_{Nominal}$ and closer to C_{Target}. On the other hand, if C_{Target} is less than $C_{Nominal}$, then adding capacitors in parallel will not help, and one or more capacitors should be added in series to get a value less than $C_{Nominal}$.

If it is determined at step 2 that C_{Target} is greater than $C_{Nominal}$ and capacitors are to be added in parallel, the process proceeds to step 6. At step 6, the algorithm determines the maximum number of capacitors J, which, when placed in parallel, results in a capacitance less than C_{Target}. Thus, if C_{Target} at this point is, for example, 7.3 pF, where $C_{Nominal} = 1$ pF, Step 6 will result in the finding that J = 7. In step 7, the number of capacitors J determined in step 6 is added to the existing value of C_{Result}, and the value of C_{Result} is updated accordingly. In the example where J = 7, 7 of the nominal value capacitors are added in parallel to the compound element being created, and the numerical value of C_{Result} is modified accordingly. If steps 6 and 7 are occurring for the first time, then C_{Result} will be these seven capacitors in series, and the numerical value of C_{Result} will be 7 pF. If this is not the first time steps 6 and 7 occur, these 7 capacitors will be added to the compound element in the appropriate place. At step 8, the value of C_{Target} is modified to reflect the addition of the J capacitors by making the new value of C_{Target} equal to the prior value of C_{Target} minus the nominal value of the added capacitors, i.e., J times the nominal capacitance of a single capacitor. Since, in this case, $C_{Nominal} = 1$ pF, the new value of C_{Target} is equal to the prior value of C_{Target} minus 7. In this example, if the prior value of C_{Target} is 7.3 pF, then the new value of C_{Target} is 7.3 minus 7, or 0.3 pF.

Table 1. Series/parallel combinations of capacitors.

Target Capacitance Value	Series/Parallel Combinations of $C_u = 280$ fF
100 fF	$(C_u + C_u + C_u + C_u \| C_u + C_u \| C_u) \| C_u + C_u + C_u$
115 fF	$(C_u \| C_u \| C_u + C_u + C_u + C_u) \| C_u \| C_u + C_u + C_u$
376 fF	$[(C_u \| C_u \| C_u \| C_u) \| (C_u + C_u) \| C_u + C_u + C_u] \| C_u$
567 fF	$[(C_u \| C_u + C_u) \| C_u + C_u] \| (C_u \| C_u + C_u + C_u) \| C_u$
1000 fF	$(C_u \| C_u \| C_u \| C_u + C_u \| C_u + C_u) \| C_u \| C_u \| C_u$

Returning to step 2, if the processor instead determines that C_{Target} is less than $C_{Nominal}$ and thus capacitors are to be added in series, the process proceeds to step 3. At step 3, the algorithm determines the maximum number of capacitors K which, when placed in series, will result in a capacitance still greater than C_{Target}. Thus, if C_{Target} at this point is, for example, 0.3 pF, again with $C_{Nominal} = 1$ pF, Step 3 will result in finding that K = 3 since three capacitors in series will have a capacitance of 0.3333 pF. In step 4, the number of capacitors K determined in step 3 is added to the existing value of C_{Result} at the appropriate location, and the value of C_{Result} is again updated accordingly.

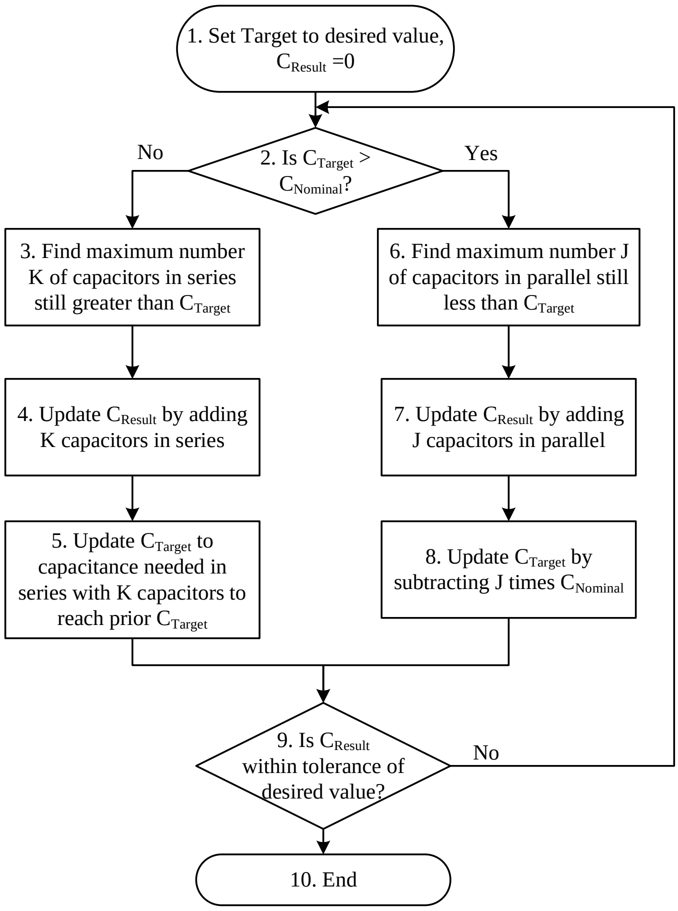


Figure 8. Flow chart of the single-value series/parallel algorithm.

At step 5, a new value of C_{Target} is set, now to the capacitance value, which, if placed in series with the K capacitors, would result in the prior value of C_{Target}. It will be appreciated that in the example given, to obtain a value of 0.3 pF with an element having an effective capacitance of 0.3333 pF, another capacitance of 3 pF must be placed in series with the 0.3333 pF capacitance. Thus, the new C_{Target} will be 3 pF.

After either step 5 or step 8, i.e., after capacitors have been added in either series or parallel and C_{Result} and C_{Target} updated accordingly, the algorithm goes to step 9 where C_{Result} is compared to the desired final capacitance value to determine if the new value of C_{Result} is within the desired tolerance of the desired compound capacitance value. If C_{Result} is close enough to the desired capacitance value, the algorithm ends at termination step 10. If C_{Result} is not close enough to the desired value, then the algorithm returns to step 2 and continues with the updated value of C_{Target}. The algorithm continues with these steps until the built-up value C_{Result} of the compound element that has been created by this process is within the desired tolerance. It has been found in practice that the algorithm will always create a compound element within any specified tolerance, i.e., the value of C_{Result} will converge on the desired total capacitance.

There are at least three advantages to building capacitors in this way. First, all the cells are identical and matched in layout; the variation of the absolute capacitance of the unit cells over process or temperature does not affect the ratios among a set of capacitors. Second, the end effects of the capacitors are not important in the ratio-based applications due to the identical unit cells. Last but not least, commercially available routing tools can be used to perform the connections; this is substantially more efficient compared to the custom layout design. In this paper, we use a group of 16 capacitors with a single value of 10 fF to build the coefficients of the multi-phase FIR filters. It covers a capacitance range from 0.625 fF to 150 fF.

The implemented CPPG contains 4 sub-FIR filters, each of which has 80 taps. Therefore, there are 400 coefficients (capacitor values) in total. Since each coefficient is constructed with 16 unit capacitor cells, a total of 6400 unit cells are needed for the filters. It is not trivial to route such a large amount of capacitors with various combinations/connections by hand. On the contrary, it is an effortless task for commercially available automatic placement and route tools to perform. The mechanism of automatic placement and routing is not in the scope of this paper.

2.4. Continuous Zero-Crossing Detectors

The continuous zero-crossing detectors (ZCDs) are high-gain amplifiers. The conventional ZCDs are implemented in the current mode logic (CML) style using one or more stages for the pre-amplification, which is neither power efficient nor area efficient [22]. In this paper, a self-biased differential ZCD is proposed without using CML, as shown in Figure 9. The current-reuse technique at the differential inputs not only reduces the power dissipation but also increases the input transconductances, thus the overall gain. Transistors M_1 and M_4, M_3, and M_6 are used as output common-mode feedbacks to set the outputs at appropriate DC voltages. Assuming that all of the PMOS transistors are identical and all of the NMOS transistors have the same size, the gain of the ZCD can be approximated as follows:

$$G = \frac{3}{2} \cdot (g_{mP} + g_{mN}) \cdot (r_{oP} \| r_{oN}) \tag{5}$$

where g_{mP} and g_{mN} are the transconductances of the input transistors M_2 and M_4, M_5 and M_7, respectively, and r_{oP} and r_{oN} are the output capacitance of the PMOS transistors and NMOS transistors, respectively.

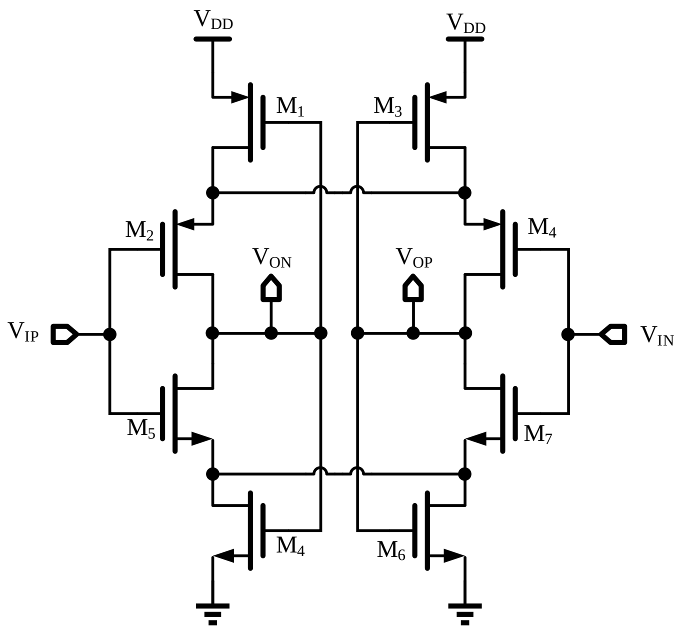
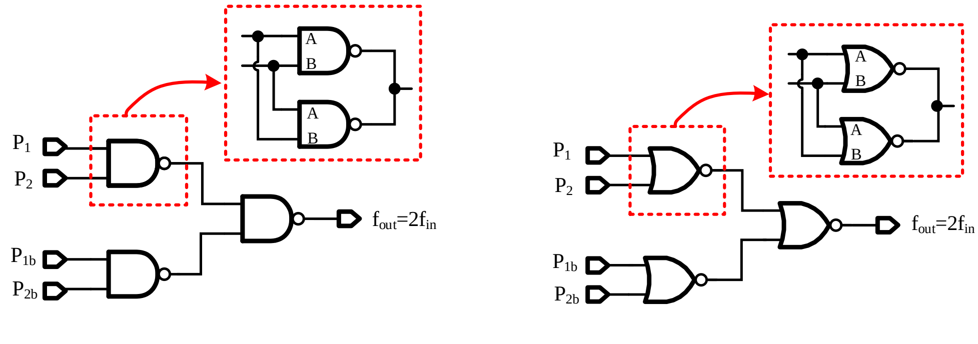


Figure 9. Self-biased zero-crossing detector circuit.

2.5. Edge Combiner

Simple logic gates can be used for edge combination by cascading serval unit edge combiners, input clock frequency can be multiplied. Figure 10 shows an example of the edge combiner where P_1 and P_{1b}, P_2 and P_{2b} are quadrature signals with a frequency of f_{in}. Due to the body effects in the series connected NMOS transistors in the conventional NAND gates, the falling edges of the edge combiner are mismatched at different input patterns, which results in the systematic error in the output clock. This issue can be addressed by adding an identical NAND gate with swapped input ports, as shown in the dashed box of Figure 10a. Similarly, the edge combiner implemented with NOR gates also has swapped input ports as shown in Figure 10b.



(**a**) (**b**)

Figure 10. Unit edge combination circuit. (**a**) Edge combiner implemented with NAND gates; (**b**) Edge combiner implemented with NOR gates.

2.6. System Schematic

The schematic of the proposed clock multiplier is shown in Figure 11. The input clock signal f_{in} will first go through the delay line and 4 capacitors arrays in CPPG to generate the four primary phases. The four primary phases will feed into 20 CSG to generate 20 sub-phase signals. These sub-phases will be compared in 20 ZCDs and then combined with multi-stage ECs. The proposed clock multiplier is an open-loop system and can only support a fixed multiplication factor of 5, but optimal output jitter performance with various input clock frequencies can also be achieved by tuning the unit delay time of the CPPG. The flowchart of the frequency tracking mechanism is shown in Figure 12. At step 1, input clock signal frequency f_{in} is defined by a control unit such as an application processor in an SoC. At step 2, since the CPPG delay line has a tuning range which is corresponding with input frequency varying from ±20% of center frequency 20 MHz, the application processor determines whether f_{in} is in the range of 20 MHz ± 20%. If f_{in} is in the right range, at step 3, the control unit will find the corresponding unit delay time in the look-up table (LUT) stored in memory. In step 4, the control unit will control the programmable delay line bias current source shown in Figure 5 to tune the bias current at the correct level to generate the appropriate unit delay time for the input clock frequency. At step 5, optimal jitter performance can be achieved through the proposed frequency tracking mechanism.

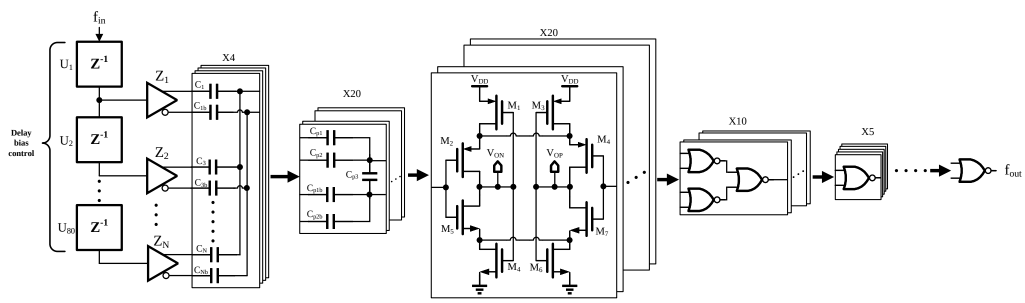


Figure 11. System schematic of the proposed clock multiplier.

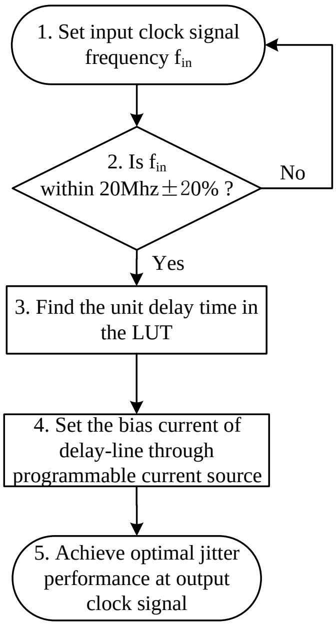


Figure 12. Working flow of frequency tracking mechanism in the proposed clock multiplier with various input clock frequencies.

3. Simulation Results

A prototype of the proposed clock frequency multiplier has been designed and laid out in a 0.18 µm digital CMOS process. Figure 13 shows the layout of the clock frequency multiplier, where the clock multiplier occupies an area of 920 µm by 1020 µm. The majority of the area is occupied by capacitor arrays that compose the CPPG and CSG. As aforementioned, although the capacitor arrays are used extensively in the design, it is an effortless task for commercially available automatic placement and route tools to perform the layout.

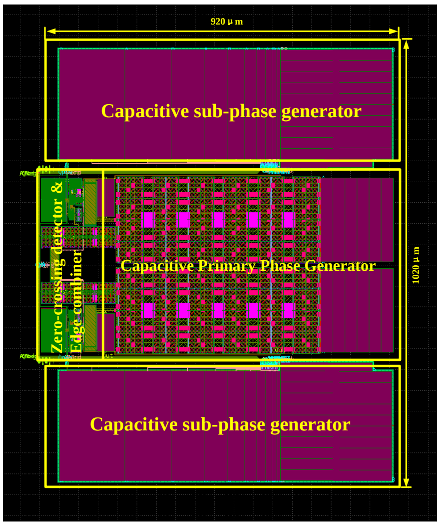


Figure 13. Chip layout of the proposed capacitive FIR-based clock multiplier.

The input 20 MHz clock signal of the proposed capacitive FIR-based clock multiplier can be various types of clock sources, such as crystal and voltage-controlled oscillators, PLLs, and DLLs. In this paper, to improve the validity of the post-layout simulation, we employ the 20 MHz input clock signal from the voltage-controlled oscillator embedded in the RIGOL-DG1022U function generator. As shown in Figure 14, we connected the output of the RIGOL-DG1022U function generator to the input of the MSO8204 digital oscilloscope. The 20 MHz output clock signal from the RIGOL-DG1022U function generator was directly captured and stored in the memory of the oscilloscope. We then exported the 20 MHz waveform data in .csv format into a USB flash disk from the oscilloscope. The waveform data was then regarded as an input signal to the extracted post-layout netlist of the clock multiplier in Smartspice simulation on a PC. Since the jitter of the RIGOL-DG1022U function generator is 6 to 7 ns [23], the 2 GHz bandwidth digital oscilloscope MSO8204 has enough bandwidth margin to capture the voltage and timing data points that contain the jitter information of the 20 MHz waveform [24].

Figure 15a shows the input/output clock signals of the clock multiplier after post-layout simulation. The input clock frequency is 20 MHz, and the output clock frequency is 100 MHz. The measured lock time of the frequency multiplier is shown in Figure 15b. The first edge of the output clock is about 175 ns delay to the input clock, which is about 3.5 clock cycles for a 20 MHz input. Figure 16 shows the simulated jitter performance of the frequency multiplier. The input clock jitter is in the range of 7 ns, and the output RMS clock jitter is reduced to about 153 ps at TT corner, 27 °C. Simulation results of the proposed clock multiplier under different corners and temperatures are demonstrated in Table 2; the robustness of the clock multiplier is maintained over different process corners and temperatures. To further verify the robustness of the proposed clock multiplier, we also performed a Monte Carlo simulation on the clock multiplier for 50 iterations with relative variation ±8% of the nominal device values, which include resistors, capacitors, and transistors. We define the output clock jitter as the output variable in the Smartspice Monte Carlo simulation to evaluate the impact of random process variations on the proposed clock multiplier. Since Smartspice will only show the mean (μ) and standard deviation (σ) of the output variable, each set of μ and σ is referred to as individual normal distribution at different temperatures and process corners. The Monte Carlo simulation results at 27 °C, −20 °C and 85 °C in different process corners are demonstrated in Table 3. The normal distribution plot of the output clock jitter at different simulation conditions is also shown in Figure 17. It can be seen that the proposed clock multiplier circuit is resilient to random device variations. The performance of phase interpolation clock multiplier circuits is also compared in Table 4.

Table 2. Simulation results of the proposed clock multiplier under different corners and temperatures.

Temperature	Parameters	TT	SS	SF	FS	FF
+27 °C	Output jitter	153 ps	172 ps	148 ps	158 ps	160 ps
	P_d [1]	5.2 mW	6.5 mW	5.7 mW	5.5 mW	5.8 mW
−20 °C	Output jitter	147 ps	167 ps	149 ps	150 ps	148 ps
	P_d [1]	4.8 mW	5.6 mW	4.7 mW	4.9 mW	5.9 mW
+85 °C	Output jitter	173 ps	188 ps	170 ps	174 ps	177 ps
	P_d [1]	6.3 mW	8.8 mW	6.6 mW	6.5 mW	8.1 mW

[1] Power dissipation of the clock multiplier when input 20 MHz clock signal and output 100 MHz clock signal.

Table 3. Monte Carlo simulation results of the proposed clock multiplier under different corners and temperatures (50 iterations with relative variation ±8% of the nominal device values).

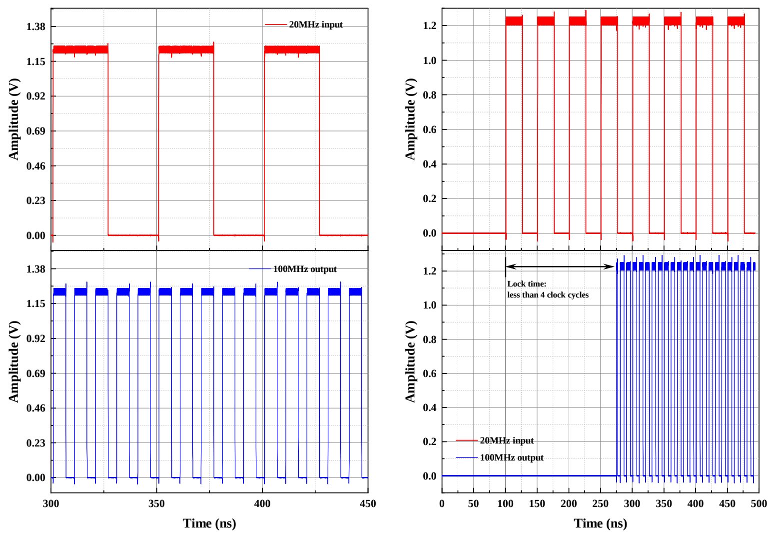
Temperature	Parameters	TT	SS	SF	FS	FF
+27 °C	Output jitter μ	160 ps	180 ps	159 ps	162 ps	172 ps
	Output jitter σ	8.2 ps	11.3 ps	8.8 ps	9.3 ps	10.2 ps
−20 °C	Output jitter μ	156 ps	158 ps	167 ps	163 ps	153 ps
	Output jitter σ	9.5 ps	8.4 ps	9.2 ps	9.5 ps	10.4 ps
+85 °C	Output jitter μ	181 ps	196 ps	182 ps	187 ps	177 ps
	Output jitter σ	12.5 ps	10.4 ps	11.2 ps	10.7 ps	11.6 ps

Table 4. Performance comparison of phase interpolation clock multipliers

Reference	Input Clock Frequency	Output Clock Frequency	Lock Time	Process	Supply Voltage	Power Consumption	Input Clock Jitter	Output Clock Jitter
[11]	156 MHz	622 MHz	1.3 cycles	250 nm	2.5 V	15 mW	3.689 ns	289 ps
[12]	25 MHz	200 MHz	–	65 nm	1.5 V	16.4 mW	25.4 ps	2.4 ps
[25]	312.5 MHz	5 GHz	–	65 nm	1.2 V	9.4 mW	–	0.55 ps
This work [1]	20 MHz	100 MHz	3.5 cycles	180 nm	1.2 V	5.2 mW	7 ns	153 ps

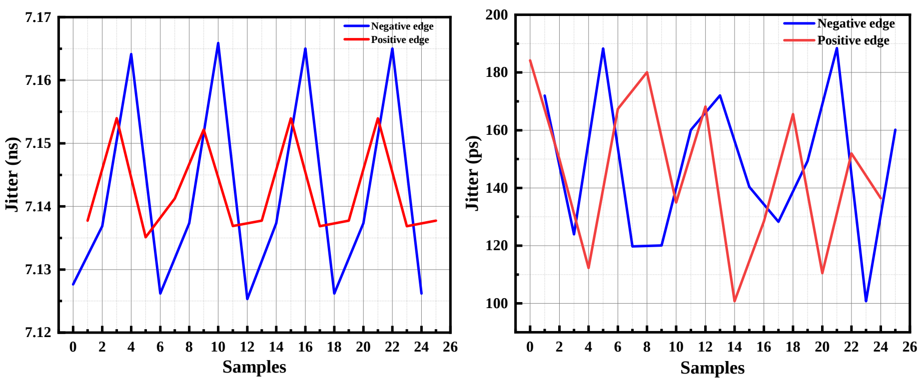
[1] Post-layout simulation results at TT corner and 27 °C.

Figure 14. Simulation setup diagram of the proposed clock frequency multiplier.



(**a**) (**b**)

Figure 15. Simulated input/output time-domain signals. (**a**) Simulated input/output clocks; (**b**) Simulated output clock lock time.



(**a**) (**b**)

Figure 16. Simulated input/output jitter. (**a**) Input clock jitter; (**b**) Output clock jitter.

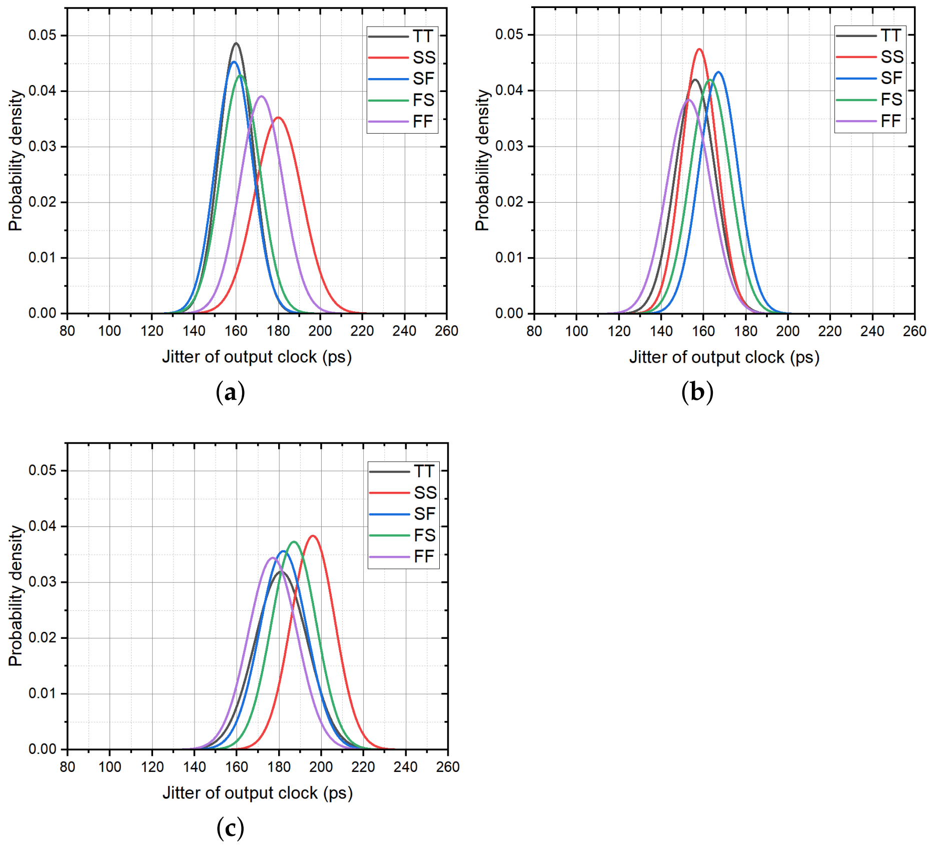


Figure 17. Graphic results of Monte Carlo simulation on the proposed clock multiplier (50 iterations with relative variation ±8% of the nominal device values). (**a**) Distribution of output clock jitter at 27 °C; (**b**) Distribution of output clock jitter at −20 °C; (**c**) Distribution of output clock jitter at 85 °C.

4. Conclusions

This paper describes a DLL/PLL-free clock frequency multiplier. Capacitive FIR filters are proposed to produce the primary phases, and a 5-capacitor network is introduced for generating arbitrary sub-phases. The capacitive FIR filters also reduce the input clock jitter. A single-value series/parallel algorithm for the capacitive networks is proposed to enable the aforementioned techniques of generating accurate and reliable primary phases and sub-phases. The major building blocks of the clock multiplier are discussed in detail. The proof-of-concept prototype is designed and laid out in a 0.18 μm digital CMOS process. The clock multiplier achieves five times the input clock frequency while reducing the input clock jitter by 33 dB at the same time. It generates the frequency multiplied clock in about 3.5 clock cycles in post-layout simulation.

As an exploration of overcoming the limitations that exist in PLL and DLL clock multipliers using novel clock phase interpolation techniques, the proposed FIR-filter-based clock multiplier provides convincing evidence to support that it is viable to achieve clock multiplication that has short locking time, low design effort, as well as low power and silicon budget without using conventional PLL and DLL architecture. Silicon verification is needed in future work to further consolidate this conclusion.

Author Contributions: Conceptualization, Z.Z. and L.Z.; methodology, Z.Z. and L.Z.; validation, Z.Z., L.Z.; formal analysis, Z.Z.; investigation, Z.Z. and L.Z.; resources, Z.Z., L.Z. and L.G.; data curation, Z.Z. and L.Z.; writing—original draft preparation, Z.Z.; writing—review and editing, Z.Z., L.Z. and L.G.; visualization, Z.Z., L.Z. and L.G.; supervision, L.G.; project administration, Z.Z. and N.Z.; funding acquisition, Z.Z. and N.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work is funded by the Shihezi University International Science and Technology Cooperation Promotion Project (Grant No. GJHZ202106).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Fan, Y.; Young, I.A. Low Power Clock Generator Design with CMOS Signaling. *IEEE Open J. Solid-State Circuits Soc.* **2021**, *1*, 162–170. [CrossRef]
2. Zhao, Y.; Memioglu, O.; Kong, L.; Razavi, B. A 56-GHz Fractional-N PLL with 110-fs Jitter. *IEEE J. Solid-State Circuits* **2023**, *58*, 57–67. [CrossRef]
3. Shin, D.; Kim, H.S.; Liu, C.c.; Wali, P.; Murthy, S.K.; Fan, Y. 11.5 A 23.9-to-29.4 GHz Digital LC-PLL with a Coupled Frequency Doubler for Wireline Applications in 10 nm FinFET. In Proceedings of the 2021 IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, CA, USA, 13–22 February 2021; Volume 64, pp. 188–190.
4. Bertulessi, L.; Karman, S.; Cherniak, D.; Garghetti, A.; Samori, C.; Lacaita, A.L.; Levantino, S. A 30-GHz Digital Sub-Sampling Fractional-*N* PLL with- 238.6-dB Jitter-Power Figure of Merit in 65-nm LP CMOS. *IEEE J. Solid-State Circuits* **2019**, *54*, 3493–3502. [CrossRef]
5. Park, H.; Sim, J.; Choi, Y.; Choi, J.; Kwon, Y.; Kim, C. A 2.4–8 GHz Phase Rotator Delay-Locked Loop Using Cascading Structure for Direct Input–Output Phase Detection. *IEEE Trans. Circuits Syst. II Express Briefs* **2022**, *69*, 794–798. [CrossRef]
6. Chang, H.H.; Liu, S.I. A wide-range and fast-locking all-digital cycle-controlled delay-locked loop. *IEEE J. Solid-State Circuits* **2005**, *40*, 661–670. [CrossRef]
7. Jung, D.H.; An, Y.J.; Ryu, K.; Park, J.H.; Jung, S.O. All-Digital Fast-Locking Delay-Locked Loop Using a Cyclic-Locking Loop for DRAM. *IEEE Trans. Circuits Syst. II Express Briefs* **2015**, *62*, 1023–1027. [CrossRef]
8. Gholami, M.; Ardeshir, G. Jitter of Delay-Locked Loops due to PFD. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2014**, *22*, 2176–2180. [CrossRef]
9. Kumaki, S.; Johari, A.H.; Matsubara, T.; Hayashi, I.; Ishikuro, H. A 0.5 V 6-bit scalable phase interpolator. In Proceedings of the 2010 IEEE Asia Pacific Conference on Circuits and Systems, Kuala Lumpur, Malaysia, 6–9 December 2010; pp. 1019–1022.
10. Sievert, S.; Degani, O.; Ben-Bassat, A.; Banin, R.; Ravi, A.; Thomann, W.; Klepser, B.U.; Boos, Z.; Schmitt-Landsiedel, D. A 2 GHz 244 fs-resolution 1.2 ps-peak-INL edge interpolator-based digital-to-time converter in 28 nm CMOS. *IEEE J. Solid-State Circuits* **2016**, *51*, 2992–3004. [CrossRef]
11. Saeki, T.; Mitsuishi, M.; Iwaki, H.; Tagishi, M. A 1.3-cycle lock time, non-PLL/DLL clock multiplier based on direct clock cycle interpolation for "clock on demand". *IEEE J. Solid-State Circuits* **2000**, *35*, 1581–1590. [CrossRef]
12. Yin, J.K.; Chan, P.K. A low-jitter polyphase-filter-based frequency multiplier with phase error calibration. *IEEE Trans. Circuits Syst. II Express Briefs* **2008**, *55*, 663–667. [CrossRef]
13. Hanumolu, P.K.; Brownlee, M.; Mayaram, K.; Moon, U.K. Analysis of charge-pump phase-locked loops. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2004**, *51*, 1665–1674. [CrossRef]
14. Hu, S.; Jia, C.; Huang, K.; Zhang, C.; Zheng, X.; Wang, Z. A 10 Gbps CDR based on phase interpolator for source synchronous receiver in 65 nm CMOS. In Proceedings of the 2012 IEEE International Symposium on Circuits and Systems (ISCAS), Seoul, Republic of Korea, 20–23 May 2012; pp. 309–312.
15. Jakobsson, A.; Serban, A.; Gong, S. A low-noise *RC*-based phase interpolator in 16-nm CMOS. *IEEE Trans. Circuits Syst. II Express Briefs* **2018**, *66*, 1–5.
16. Parks, T.W.; Burrus, C.S. *Digital Filter Design*; Wiley-Interscience: Hoboken, NJ, USA, 1987.
17. Williams, A.B.; Taylor, F.J. *Electronic Filter Design Handbook*; McGraw-Hill Education: New York, NY, USA, 2006.
18. Schlichthärle, D. *Digital Filters*; Springer: Berlin/Heidelberg, Germany, 2000.
19. Hinamoto, T.; Lu, W.-S. *Digital Filter Design and Realization*; CRC Press: Boca Raton, FL, USA, 2022.
20. Bhattacharya, B.; Bhattacharyya, S.S. Parameterized dataflow modeling for DSP systems. *EEE Trans. Signal Process.* **2001**, *49*, 2408–2421. [CrossRef]
21. LeGuernic, P.; Gautier, T.; Le Borgne, M.; Le Maire, C. Programming real-time applications with SIGNAL. *Proc. IEEE* **1991**, *79*, 1321–1336. [CrossRef]
22. Shin, S.K.; You, Y.S.; Lee, S.H.; Moon, K.H.; Kim, J.W.; Brooks, L.; Lee, H.S. A fully-differential zero-crossing-based 1.2 V 10b 26MS/s pipelined ADC in 65 nm CMOS. In Proceedings of the 2008 IEEE Symposium on VLSI Circuits, Honolulu, HI, USA, 18–20 June 2008; pp. 218–219.
23. RIGOL Technologies. Available online: https://int.rigol.com/products/DG_detail/DG1000 (accessed on 29 August 2019).

24. RIGOL Technologies. Available online: https://int.rigol.com/products/detail/MSO8000 (accessed on 1 November 2021).
25. Gautam, R.; Bandarupalli, J.D.; Saxena, S. A 2.5–5 GHz Injection-Locked Clock Multiplier with Embedded Phase Interpolator in 65 nm CMOS. In Proceedings of the 2020 IEEE International Symposium on Circuits and Systems (ISCAS), Sevilla, Spain, 10–21 October 2020; pp. 1–5.