An All-Digital Phase-Locked Loop (ADPLL)-Based Clock Recovery Circuit

Terng-Yin Hsu, Bai-Jue Shieh, and Chen-Yi Lee

***Abstract*— A new algorithm for all-digital phase-locked loops (ADPLL) with fast acquisition and large pulling range is presented in this paper. Based on the proposed algorithm, portable cell-based implementations for clock recovery with functions of a frequency synthesizer and on-chip clock generator are completed by standard cell. These modules have been designed and verified on a 0.6-μm CMOS process. Test results are summarized as follows: 1) the proposed ADPLL can satisfy full locked bandwidth and fast acquisition within one data transition; 2) the on-chip clock generator can generate any target clock rate $f_{\rm clock}$; and 3) the function of nonreturn-to-zero clock recovery has a maximum $f_{\rm clock}/4$ recovering capability with a locking range of ($\tau_{\rm input} \pm \tau_{\rm input}/2$), where $\tau_{\rm input}$ is the input period.**

***Index Terms*— All-digital phase-locked loop (ADPLL), clock recovery, frequency synthesizer, phase-locked loop.**

I. Introduction

PHASE-locked loops (PLL's) have been developed for a long time. The major concept is used as an oscillator to lock or track input signals in both phase and frequency [1]. In communication applications, PLL's can make local clocks synchronous with other signals. Because digital designs are popular presently, and they fit in baseband designs, digital PLL's (DPLL's) are widely used in many modern applications, such as ethernet, asynchronous transfer mode, wireless local-area network, microprocessors, digital signal processing, etc. In digital communications, information is expressed by a series of 1's and 0's. To process the received data correctly, it is necessary to synchronize a local clock to received data. However, some design issues must be taken into account when DPLL is to be used as clock recovery, such as phase detection, frequency accumulation, initial constraints, etc. The key characteristics of the clock recovery are 1) pulling range and 2) lock-in time ($T_{\rm lock}$). As a result, the major goal in this paper is to design a digital PLL-based clock recovery with both large pulling range and short lock-in time.

For advances and improvements of digital very-large-scale-integration circuits (VLSI's), all-digital phase-locked loops (ADPLL's) have good abilities to meet the requirements of computer and communication applications. The *portable*[1] design is an important issue in digital VLSI. A problem of portable ADPLL's is the source of on-chip high-speed clocks. Theoretically, this high-speed clock should be fast enough and act as a reference clock. When ADPLL's operate in high-data-rate environments, a number of issues must be taken into account, such as power dissipation, logic propagation delay, process variation, and so on. Hence the characteristics of on-chip high-speed clocks must be limited for a target process and flexible enough to meet design requirements. It means that an on-chip high-speed clock must be controllable to operate at a special rate.

Many PLL/DPLL-based designs have been developed. In [2], a nonreturn-to-zero (NRZ) timing recovery with a digital phase detector (PD), analog loop filter (LF), and voltage-controlled oscillator (VCO) are introduced for band-limited applications. In [3] is shown a fully integrated CMOS frequency synthesizer with an analog LF and current-controlled ring oscillator. Reference [4] presents several design techniques to improve the performance of a PLL with an analog LF and current steering amplifier ring oscillator. They are usually based on conventional design techniques. In [5], an idea of implementing a variable-bandwidth DPLL by finite state machine (FSM) is described and simulated. It can be completed by all-digital implementations; however— fully custom layout still has to be performed.

In this paper, an ADPLL-based clock recovery is introduced here, which has already met requirements of large pulling range, short lock-in time, high data rate, high operating frequency, and portable solution. It is different from conventional DPLL's in tracking and locking mechanism. In implementations, all parts belong to portable cell-based designs and have been verified on silicon using *an in-house* 0.6-μm CMOS single-poly, triple-metal (SPTM) cell library. According to testing results, the proposed clock recovery can operate at a maximum rate of $f_{\rm clock} = 165$ MHz to recover $f_{\rm clock}/4$ NRZ data within one data transition and with a locked bandwidth of ($\tau_{\rm input} \pm \tau_{\rm input}/2$), where $\tau_{\rm input}$ is the input period. An extra frequency-synthesizer module, cascaded with NRZ clock recovery, can produce expected frequencies whose maximum output frequency is equal to the fastest on-chip clock. The on-chip clock generator can generate any number of frequencies with a maximum output rate of 165 MHz.

The rest of this paper is organized as follows. The proposed algorithm and architecture of ADPLL is first addressed in Section II. The ADPLL-based clock recovery with an extra frequency-synthesizer module and on-chip clock generator for portable cell-based design is presented in Section III.

Manuscript received September 28, 1998; revised April 27, 1999. This work was supported by the National Science Council of Taiwan, R.O.C., under Grant NSC88-2215-E-009-068.

The authors are with the Department of Electronics Engineering, National Chiao Tung University, Hsinchu, Taiwan 300 R.O.C. (e-mail: cylee@cc.nctu.edu.tw).

Publisher Item Identifier S 0018-9200(99)06161-2.

[1] By portable, we mean that the design can be done by hardware description language (HDL) and targeted to different CMOS processes through synthesis tools.

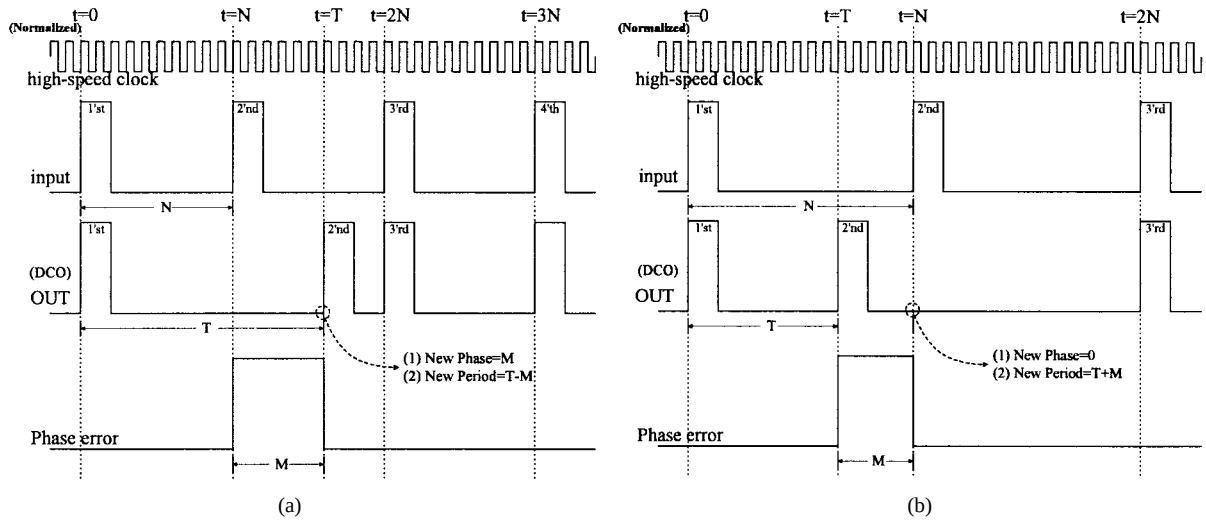


Fig. 1. Basic concept for estimating input's phases and frequencies within two input edges in two different conditions of period: (a) T(DCO) $> N$(input) and (b) T(DCO) $< N$(input).

Implementations and test results of the proposed solutions are then described and discussed in Section IV.

II. The Proposed ADPLL

A. Algorithm Description

The general concept of ADPLL and its two basic structures can be found in [6]. In practical situations, input information is very important for tracking or recovering desired signals correctly. To extract accurately and track immediately input information, it is necessary to estimate both phases and frequencies of input signals continuously. As a result, the object of the proposed ADPLL algorithm is to achieve fast acquisition and large pulling range.

It is assumed that a high-speed clock is used as reference timing, and all signals must be referred to this clock. The periods of a digital-controlled oscillator (DCO) and input signal are equal to $T{\cdot}\tau'$ and $N{\cdot}\tau'$, where T and N are constant and τ' is the period of high-speed clock (reference). After normalization with τ', the new (normalized) periods of DCO and input are T and N, respectively. The proposed algorithm is illustrated in Fig. 1(a) and (b) for two conditions. Fig. 1(a) shows the case of the DCO's frequency (1/period) which is slower than input signals, i.e., T(DCO) $> N$(input). Fig. 1(b) shows the other case of the DCO's frequency, which is faster than the input signal, i.e., T(DCO) $< N$(input). Note that all values are referred (normalized) to a high-speed clock τ', and all processes are under normalized conditions. The major problem is how to adjust DCO accurately to follow the input signal based on received data information.

Assume that T and N stand for normalized periods of the DCO and input signal, respectively. It is shown that N is equal to $(T - M)$ for estimating correct input information in Fig. 1(a), where M ("*M times*" *period of high-speed clock*) is the normalized phase error between the DCO and input. At $t = 0$ (time), the system starts to synchronize phases for the first valid edge of the input and DCO output. During (time) N to $2N$, a phase error M is detected. Then the correct phase for the second edge of DCO is equal to the phase error, and the correct period is determined by comparing M and T. In Fig. 1(b), tracking procedures for $N = (T + M)$ are similar to the above case, but the correct phase of DCO is "0," not "M." If there is a special case of $T > 2N$, a phase error will exceed 2π, and the phase of DCO becomes undecidable. To solve this problem, the system must resynchronize again with the previous estimation $(T - M)$. Thus, the correct phase and period are estimated for setting DCO to reach minimum errors.

Mathematically, a correct phase for the second edge of DCO corresponding to input is M, and the (normalized) period is equal to $(T - M)$ for $T > N$. Hence at time $>2N$, input and DCO output are completely synchronous in both phases and frequencies. For the case of $T < N$, it is the same as the above discussions for $N = (T + M)$, and the phase of DCO equals zero. A new period N is calculated by both phase error M and previous result T. Therefore, the lock-in time includes two conditions—DCO output, which is slower than input, and DCO output, which is faster than input. For the first case, we need one DCO normalized period T to determine the correct phase and frequency. Thus, the lock-in time of the ADPLL is equal to T; however, two clocks have the same phase at $2N$, as shown in Fig. 1(a). For the case of $(N > T)$, the correct tracking period is equal to $(T + M)$ and can be completed by one input normalized period N. Thus, the lock-in time is equal to N, and the two clocks have the same phase at $2N$, as shown in Fig. 1(b). Hence the lock-in time T_{lock}, which is defined by the time needed to determine both correct phase and frequency, can be written as follows:

$$T_{\text{lock}}|_{\text{normalization}} = \max\{N, T\}. \tag{1}$$

It is important to select suitable parameters for DCO's to reduce the lock-in time. A flowchart is shown in Fig. 2. Note that the proposed algorithm does not need to set accurate time, and the dynamic range of ADPLL is not limited by initialization. This is because both input phase and frequency are already available in ADPLL. In other words, characteristics of the proposed algorithm are irrelevant to input signal.

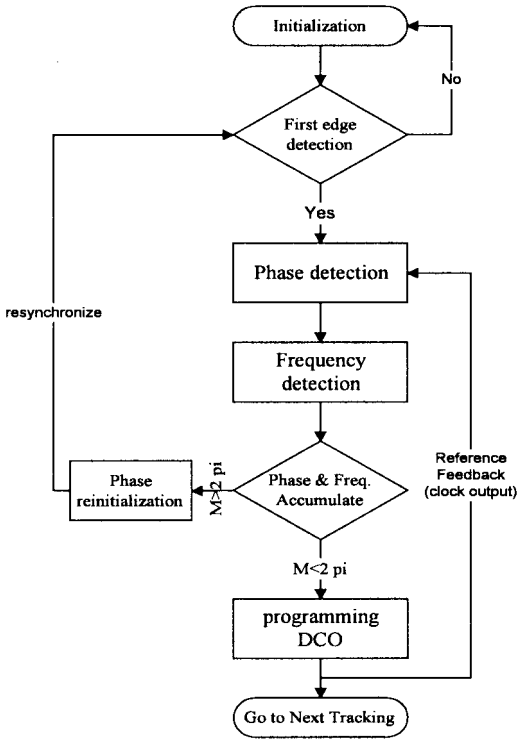


Fig. 2. Flowchart of the proposed ADPLL algorithm.

B. The Proposed Architecture

A block diagram of the proposed ADPLL is shown in Fig. 3. There are six major functional blocks, namely:

1) *differentiator with first edge detection* (DFED), whose function is to generate differential pulses [6] for increasing resolutions and detect the first edge from input signals to meet synchronous conditions, as indicated in Fig. 1;
2) *phase detector* (PD), which is a JK flip-flop [6] to indicate phase differences between differential pulses and DCO output;
3) *filter counter*, which is used to determine the phase errors of two signals by referring to a high-speed clock;
4) *phase and frequency estimator*, which is used to measure new phases and frequencies based on original information and new phase errors;
5) *programmable DCO*, which is used for generating a target signal;
6) *high-speed clock generator*, which is used as an on-chip reference timing for controlling and synchronizing all modules to avoid timing errors.

To fit the above requirements, a programmable DCO with adjustable phase and frequency is required to operate in jumping conditions. In other words, it must be able to change instantaneous phase and operating frequency at any controlled cycle to minimize both phase and frequency errors. Based on this DCO, the proposed ADPLL can dynamically determine instantaneous phase and frequency to correctly track input signals in minimal time. For practical situations, dynamic control is necessary to compensate for noise effects and process variations to make ADPLL more accurate in tracking. The operation for this architecture is explained as follows.

After initial settings are loaded, the first input valid edge must be detected by DFED to fit a synchronous condition, as shown in Fig. 1. By using the first valid edge to trigger all modules for extracting input information, the phase errors (M) are estimated by *filter counter* to calculate the difference between the second valid edge of the input and DCO output. This phase error (M) is used to determine a target period and phase for the DCO. If the phase error is greater than 2π (or $M > N$ in Fig. 1), the DCO must be stopped to make correct relations of phases until the next input edge. Last, both correct phase and period (frequency) are determined and fed into the DCO to generate synchronous signals for the next cycle. These procedures run once per valid input transition to ensure ADPLL with minimum phase and period errors. In a noisy environment, both phase and period of input signals change randomly, and all noisy phenomena can be classified into two cases. One is input phase lagging DCO output and the other is input phase leading DCO output. In both cases, the proposed ADPLL estimates new phase and period to minimize errors during tracking input signals. Although the ADPLL becomes stable with any initial setting or any condition, the lock-in time is always proportional to initial settings, as given in (1). The operating speed of this architecture is limited by the critical path from *filter counter to phase and frequency estimator*, as shown in Fig. 3. Note that both phase and period estimations need one high-speed clock. In addition, input tracking needs at least one high-speed clock. Hence the maximum tracking frequency is equal to $f_{\text{clock}}/3$ (*without differentiator*).

III. Clock Recovery

A. Kernel of the Clock Recovery

1) Algorithm: The purpose of clock recovery is to search a correct timing for input signals, such as NRZ, RZ, and so on, because these signals do not include timing information in their bitstream. Theoretically, a series of 1's and 0's of NRZ or RZ always have some uncertain problems of received phases and frequencies. To overcome these problems, DCO's are limited to certain finite dynamic ranges in phase and frequency jumping for clock recovery. To recover a NRZ signal, a differentiation process is often requested to improve resolution. Based on the proposed ADPLL algorithm, constraints for phase and frequency are added into systems to avoid uncertain problems and discussed below.

To transmit more than two 1's (or 0's), the maximum phase error of these 1's is at least 2 $\times$ (input period). The detection becomes failed in DCO without constrained dynamic ranges of phases and frequencies. Then the frequency of DCO cannot be adjusted over one input period because a minimum number of continuous 1's (or 0's) is two. The limitations of pulling up for calculating new frequencies are similar to the case of pulling down. Hence for equal probability of phase errors in both pulling-up and pulling-down cases, the maximum adjusted region is $\pm$(input_period)/2. According to the above controlled constraints, the dynamic-control range of DCO's is

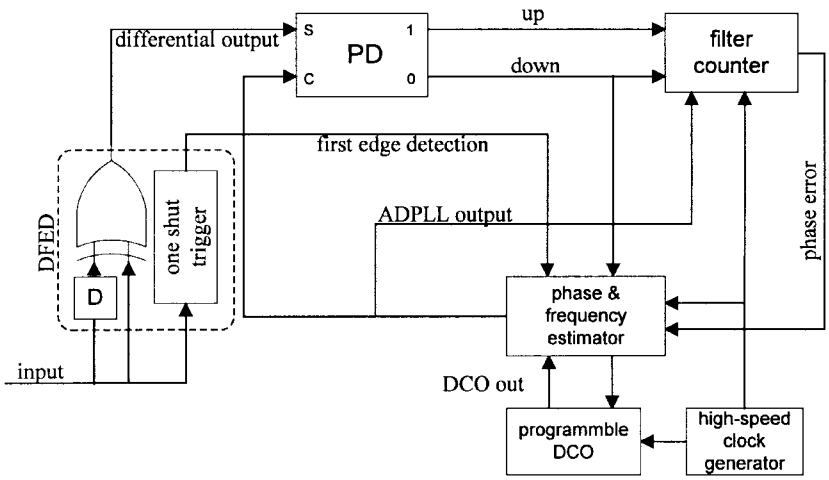


Fig. 3. Block diagram of the cell-based ADPLL architecture.

equal to $\{\tau_{\text{input}}/2, 3\tau_{\text{input}}/2\}$ or $\{2f_{\text{input}}/3, 2f_{\text{input}}\}$, where τ_{input} is the input period, and $f_{\text{input}} = \tau_{\text{input}}^{-1}$. Both stability and bandwidth must be taken into account and traded off in designs. Two detection constraints of phase errors are included in the phase and frequency estimator to ensure systems operate correctly even having errors caused by phase uncertainty of NRZ signals. Both phases and frequencies of DCO's can be accurately judged to prevent failures of locks during series of 1's or 0's. This algorithm can be restored only after one data transition.

For conventional designs, it is difficult to combine both clock recovery and a frequency synthesizer into a single system. In some applications, if a system includes several modules to process received information in parallel and the final data must be combined, multiple clocks synchronous to the recovered data clock are often needed to control data paths (such as code-phase multiplexed algorithms for direct-sequence spread spectrum [7]). Since the input period (frequency) can be extracted by a phase and frequency estimator, it is possible to directly make use of the information to generate any number of frequencies. To eliminate the instability of a feedback loop in a conventional frequency synthesizer, an extra module cascaded with clock recovery, namely, a *multiple-frequency generator,* is introduced to realize the above concept. A flowchart of the NRZ clock recovery with a frequency synthesizer is shown in Fig. 4. By using such an extra module, it is able to generate any number of frequencies without having influence from the divided-feedback loop. For taking the finite-precision issue into account

$$T_{\text{New}} \neq \left\lfloor \frac{T_{\text{clock\_recovery}}}{k} \right\rfloor$$

the function of the extra module is modified to minimize processing errors by

$$T_{\text{New}} = \left\lfloor \frac{T_{\text{clock\_recovery}} + \frac{k}{2} + 1}{k} \right\rfloor \tag{2}$$

where k is a constant (if we want to approach $f_{\text{New}} = kf_{\text{clock\_recovery}}$).

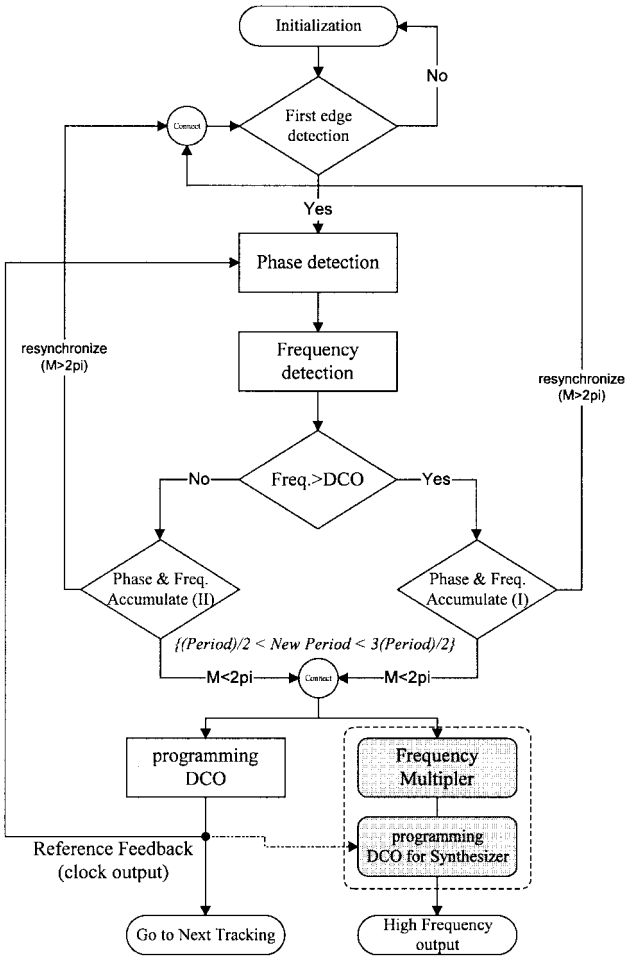
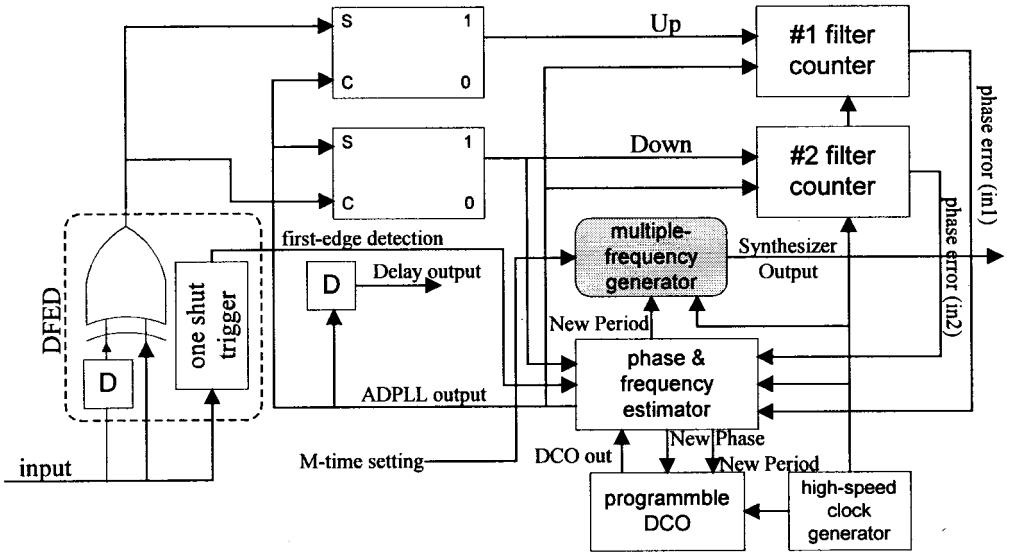
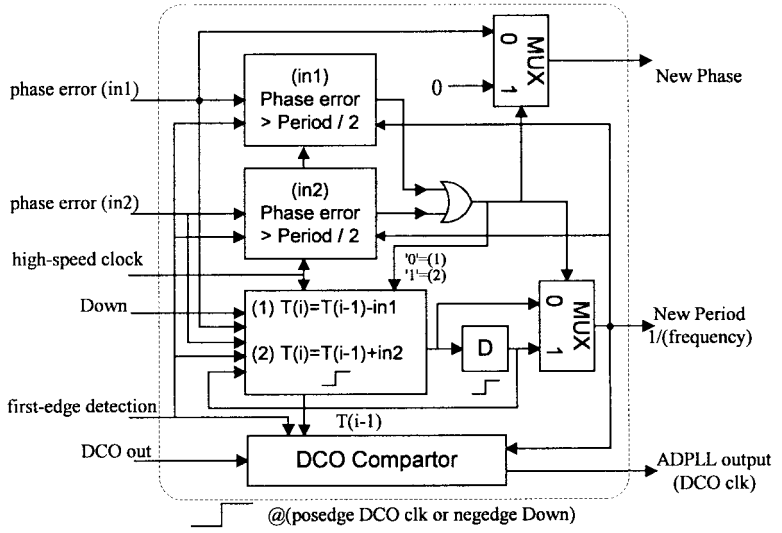


Fig. 4. Flowchart of the NRZ clock recovery algorithm with a function of the synthesizer, where the dotted line indicates extra blocks for the frequency synthesizer.

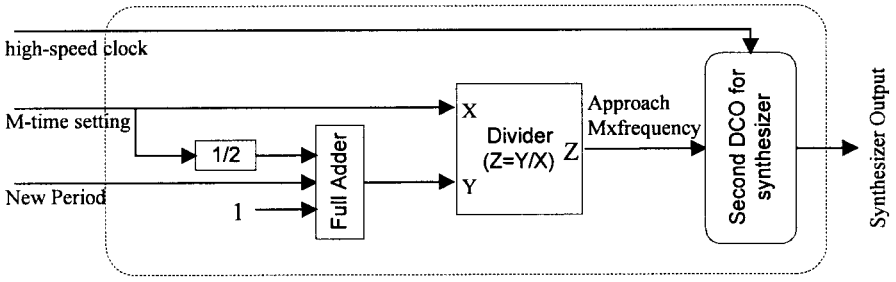
2) Architecture: A block diagram of the proposed clock recovery is shown in Fig. 5(a). In practical situations, it is essential to adjust the output of clock recovery with a fixed or programmable phase delay to correctly guarantee output timing [6]. Hence the output of clock recovery is set to delay



(a)



(b)



(c)

Fig. 5. Application example of NRZ clock recovery: (a) block diagram of NRZ clock recovery with a function of the synthesizer, (b) structure of phase and frequency estimator, and (c) structure of multiple-frequency generator.

D phases, corresponding to numbers of the high-speed clock. In other words, the output can also be corrected within D input-phase errors (jitter).

In Fig. 5(a), seven major functional blocks form the complete clock recovery. They are DFED, PD, up/down filter counter, phase and frequency estimator, high-speed clock, and programmable DCO. Due to the limitations of unpredictable phases, it cannot use conventional three-state PD's [1] to detect input phases. Compared to Figs. 3 and 5(a), the main differences are the structure of the phase and frequency estimator and filter counters. Up and down channels are separate to measure different phase errors and uncertainty problems. The

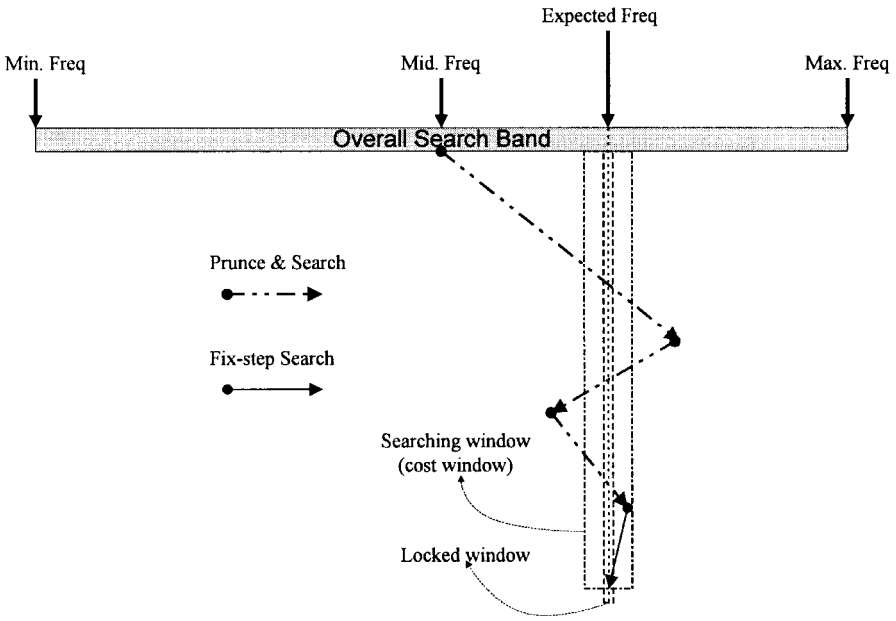


Fig. 6. Searching flow of the proposed clock generator.

most important and complex module of clock recovery is the phase and frequency estimator used to determine lock-in time, dynamic ranges, stability of phase uncertainties, jitters, etc. The structure of the phase and frequency estimator is shown in Fig. 5(b). In estimating new phase and period (frequency), the function of clock recovery is similar to ADPLL. Both phase error and frequency estimations are limited to {maximum $\tau_{\text{input}}/2$}. With this assignment, it can prevent an overlock event during any number of series of 1's or 0's within received data streams in a noisy environment. All tracking procedures are recycled to make information correct during each valid transition. Because the bandwidth of DCO's is limited within a certain range proportional to input signals, the initialization must be considered carefully to prevent recovering process failure. Generally, the data rate of input signals is known, so it is not difficult to meet these constraints, as discussed in Section III-A1. In implementations, two high-speed clocks are requested to estimate correct phases and periods, and at least two high-speed clocks must be allocated for phase jitter. So, the overall operating frequency is equal to *the maximum* $f_{\text{clock}}/4$. In Fig. 5(c), the structure of the multiple-frequency generator is mainly based on a digital constant divider and adder to calculate the appropriate period (frequency). After the correct information of received signals is estimated by clock recovery, the period (frequency) information is extracted into the multiple-frequency generator for achieving functions of frequency synthesizers. By this scheme, the output of a normal DCO and synthesizer are separated, and both the clock recovery and frequency synthesizer can connect together. Compared to conventional designs, the major difference is the multiple-frequency generator used to approach an expected period (k-*time frequency*) for the second DCO. Both normal and k-frequency (multiple frequency) DCO's are independent, but are synchronous to each other. The normal DCO is used to lock received signals, and a multiple-frequency (second) DCO is applied to approach k-time frequency.

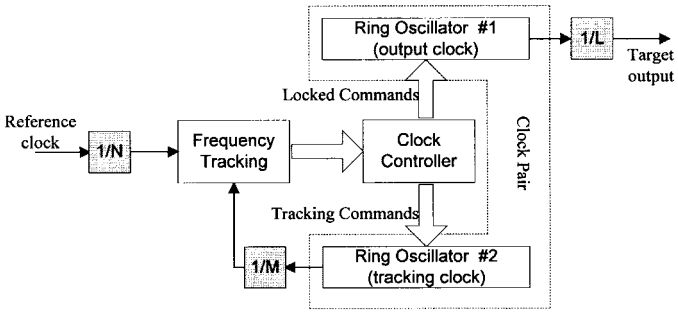


Fig. 7. Block diagram of cell-based clock generator.

B. Clock Generator Submodule

The above approach essentially needs a high-speed clock as a reference for tracking. To provide a full cell-based integration solution, it is necessary to generate this high-speed clock on chip.

1) Algorithm: An algorithm for digital cell-based oscillators is presented below to meet our requirements for full integration. Two major functional blocks are identified, namely, a frequency controller and ring-oscillator pair. Their functions are to search and control the output clock to meet design specifications and then to generate target clocks.

A default clock generated by a ring-oscillator pair is compared to an input reference for checking whether its frequency error is within a locked window. The compared result is then fed into a frequency controller to determine new controlled status or frequency-search commands for a ring-oscillator pair to update the output clock. These procedures are repeated until a target clock can synchronize to an input reference. To ensure that these search procedures are accurate and efficient, the frequency-search algorithm includes two stages: a coarse search based on a "prune-and-search" algorithm [8] and a fine search based on a "fix-step" algorithm. Two cost functions for search and lock-in processes are derived to determine

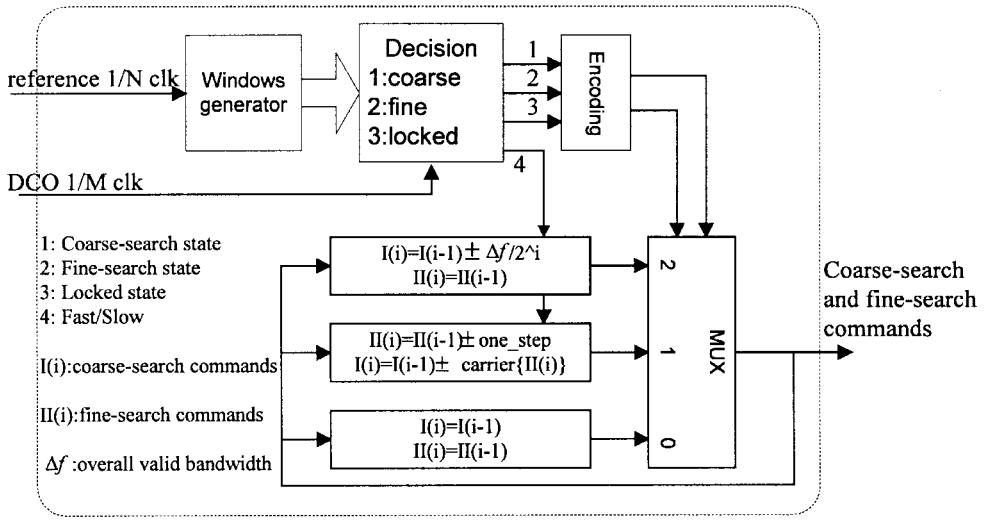


Fig. 8. Structure of frequency-tracking module in on-chip clock generator.

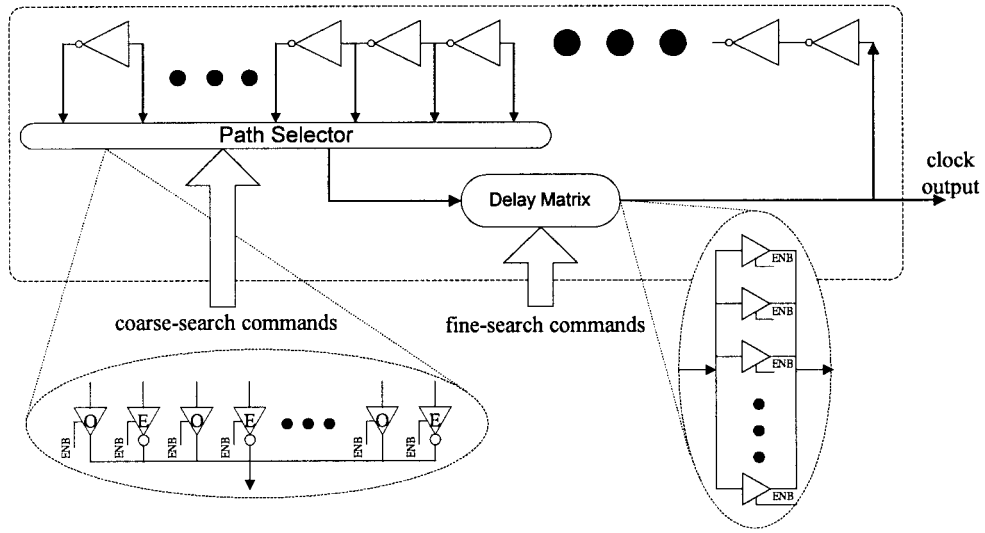


Fig. 9. Structure of cell-based digital-controlled oscillator.

which scheme should be applied to search a target frequency and locked status. If frequency errors are higher than both search and locked thresholds, a coarse search is activated to estimate a nearby frequency. On the other hand, a fine search is only activated when frequency error is between the search and locked thresholds. By properly assigning these two thresholds, the search performance and output resolution can be improved, such as the search time and frequency error. The search flow is shown in Fig. 6. It is assumed that a default frequency is set to {(min frequency + max frequency/2)}, denoted by Mid. If a target frequency is higher than Mid and its frequency error is greater than the threshold or out of a cost window, the default frequency must move from Mid to a new frequency {Mid + (max frequency min frequency)/2} based on a prune-and-search scheme. If its frequency error is smaller than the threshold or within the cost window, the search algorithm is switched to the "fix-step" fine-tune stage. Because the threshold is fixed, the worst case time complexity of fine search $T_{\mathrm{Fix}}(K)$ is a constant. Hence the worst case time complexity [8] is given below

$$\begin{aligned} T(n) &= T_{P\&S}(n) + T_{\mathrm{Fix}}(K) \\ &= O(\log_2 n) + O(C) \\ &= O(\log_2 n) \end{aligned} \tag{3}$$

where

$$\begin{aligned} T_{P\&S}(n) &= T_{P\&S}\left(\frac{n}{2}\right) + P(n) \\ &= T_{P\&S}\left(\frac{n}{2}\right) + O(1) \\ &= T_{P\&S}\left(\frac{n}{2^k}\right) + k + 0 \\ &= O(\log_2 n). \end{aligned}$$

Note that C is a constant, $O(C) = 0$, $P(n)$ is the complexity of the last step, and $T_{P\&S}(n)$ is the complexity of the prune-and-search algorithm. The overall time complexity for the condition of a target frequency slower than Mid is also equal to (3). In this way, the search process can be implemented simply by a digital design technique.

2) Architecture: A block diagram of cell-based clock generators is shown in Fig. 7. It looks like traditional designs in the searching loop; however, an independent inverter path is added to generate a target clock. Three major functional modules are allocated in the on-chip clock generator:

1) a clock pair—clock source for search and output;
2) a frequency tracking for searching target frequency and determining conditions of clock pair;
3) a clock controller to decide correct controlled timing at changing commands.

The structure looks like a conventional frequency synthesizer with an extra oscillator for target clock, but the searching mechanism is completely different. In Fig. 7, a *divider* N is used not only to slow down reference frequency but also to generate controlled timing for searching frequency. Since propagated delays have to be taken into account, overall operations are divided into two types: 1) an active region for searching a target frequency and 2) a preset region for restoring commands. Due to the physical constraints of digital VLSI, such as setup time and hold time, the decisions of frequency search must be determined carefully. Hence, in order to prevent timing violation, a *divider* M is applied to extend decision regions. In this way, the cost or threshold can be determined without a timing violation. To compensate for temperature and supply-voltage variations, the search procedure must be a *recycled action* (*continuous*), even if a target clock is found. The locked and tracking commands in Fig. 7 include both coarse-search and fine-search commands. The structure of frequency tracking is shown in Fig. 8. A cell-based digital-controlled oscillator is shown in Fig. 9. It is a kind of ring oscillator with n stages, whose characteristics are determined by a inverter. A free-running frequency is determined by the number of inverters. Note that the dynamic controlled range is determined by the number of path selectors, and frequency resolution is dependent on scales of a delay matrix. Both a path selector and a delay matrix are used to perform coarse and fine searches, and the resolution of ring oscillators is decided by the minimum scale of a delay matrix. It is important for systems not only to maintain a stable clock but also to ensure minimum frequency error. Hence two equivalent cell-based DCO's are allocated within a clock pair: one for frequency searches and the other for a target clock.

To ensure that the proposed architecture can generate any target frequency, both temperature and voltage variations have to be taken into account. From [9], it is shown that the output frequency of inverter chains is dependent on $\tau_{\rm PHL}$ and $\tau_{\rm PLH}$ of inverters (inverter delays). However, inverter delays are functions of temperature, supply voltage, loading capacitance, etc. For simplification without losing generality, it is assumed that $V_{\rm TN} = |V_{\rm TP}| = V_T$ and $k_n = k_p$ for $\tau_{\rm PHL} = \tau_{\rm PLH}$. In practical applications, the output frequency can be allowed a certain frequency error, denoted by Δf. That is, $f_{\rm output} = f_{\rm target}(1 \pm a\%)$, where $a\%$ is a percentage of frequency error, and $\Delta f = f_{\rm target} \cdot a\%$. Hence, when the output frequency is equal to (or less than) $f_{\rm target} \pm \Delta f$, the size of a locked window must be smaller than frequency error to meet output specifications. Based on Fig. 9, the output

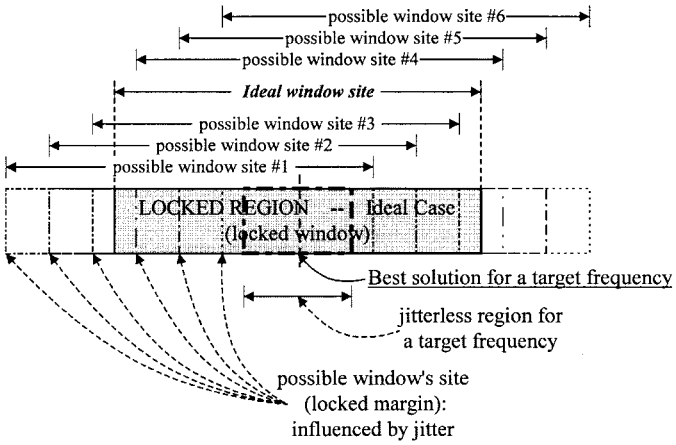


Fig. 10. Relationship among jitter, target frequency, and possible sites of a locked window.

TABLE I
GATE COUNT OF EACH SYNTHESIZED MODULE IN CLOCK RECOVERY

Module	*Gate Count*
Clock Recovery (Overall)	4393
Filter Counter (#1 and #2)	132 * 2
DFED	42
Phase & Frequency Estimator	830
Programmable DCO	100
Others (delay, buffer, JK flip-flop)	68
Multiple-Frequency Generator	842
On-Chip Clock Generator	2247

frequency of cell-based digital-controlled oscillators is

$$f_{\rm target} = \frac{1}{\tau_{\rm target}} \tag{4}$$

where

$$\tau_{\rm target} = 2\left[n \cdot \tau_{\rm PHL} + \left(\tau_{\rm PHL} + \frac{\tau_{\rm PHL}}{K} \cdot m\right)\right] + C \tag{5}$$

and C is a constant for a free-running frequency, n is the total number of inverters

$$\left[\tau_{\rm PHL} + \frac{\tau_{\rm PHL}}{K} \cdot m\right]$$

is the model of the delay matrix, and it represents the delay of the mth path in K total delay paths. In other words, the delay matrix owns K paths to pass signals with different delays, and relative delays of each path are almost linear. Note that (5) must satisfy an inequality of frequency errors as indicated below

$$\frac{1}{f_{\rm target} + \Delta f} \le 2\left[n \cdot \tau_{\rm PHL} + \left(\tau_{\rm PHL} + \frac{\tau_{\rm PHL}}{K} \cdot m\right)\right] + C \le \frac{1}{f_{\rm target} - \Delta f}. \tag{6}$$

In (6), for any given $\Delta f \neq 0$ and $\tau_{\rm PHL}$, we can find a K to satisfy the above constraints. The larger a dynamic range of n is, the wider an output bandwidth is. For increasing K in designs, resolutions of cell-based digital-controlled oscillators can be improved. To satisfy (6), both n and m are controlled by a frequency-tracking module to search a proper combination based on prune-and-search and fix-step algorithms.

TABLE II
FEATURES OF EACH MODULE USED IN CLOCK RECOVERY

Module	*Frequency*	*Pulling Range*	*Acquisition*
Core of Clock Recovery	*Max 41Mbps NRZ at 165MHz*	*Max. (82MHz~27.33MHz)*	*one data transition*
Frequency Synthesizer	*Max 165MHz at 165MHz*	*(same to clock recovery)*	*one data transition*
Clock Generator Part	*Max 165MHz (Jitter: RMS 11.42ps / P_kP_k 74ps)*	*Auto-Scan (full bandwidth)*	*$a*log_2 n$ (a : Constant)*

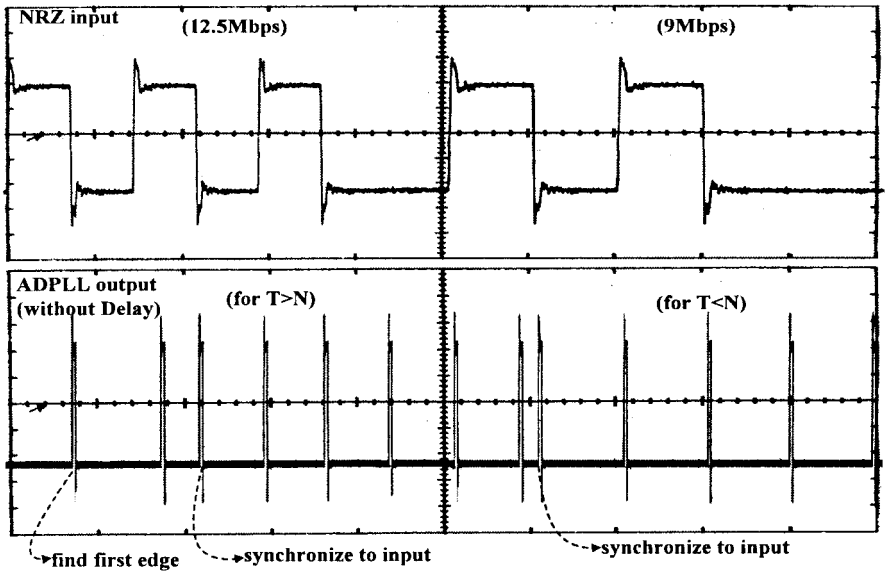


Fig. 11. Measured result of clock recovery at 125-MHz high-speed clock/(12.5- and 9-Mbps) NRZ input.

From the above description, it is shown that effects of temperature and supply voltage can be eliminated by proper selection of n, m, C, and K. Practically, a reference clock is not ideal (it includes some frequency jitter) and influences target frequency. According to the above description, the frequency-tracking module uses a locked window to make a decision. Although the site of a locked window is also influenced by jitter, the locked status of a certain target frequency is not out of locked regions (if a locked window is large enough). Generally, the size of the locked window is at least double input jitter. As a result, influences of jitter can be decreased because the target frequency is located at the window's center, as shown in Fig. 10.

IV. IMPLEMENTATION AND DISCUSSION

The clock recovery with a function of frequency synthesizer and an on-chip clock generator are fabricated in a 0.6-μm CMOS SPTM process. First, the whole design (including all functional modules) is described by Verilog-HDL with timing information from an in-house target standard-cell library. Then Verilog source codes are synthesized to generate gate-level netlists and schematics for further simulations and verifications. The synthesis result of each module is shown in Table I. These gate-level netlists are verified with original codes to check their behaviors and timing. In addition, the cost and timing of whole systems must be carefully considered to meet requirements in this step. After timing and functions have been verified correctly, we use CADENCE/OPUS to do physical designs. To minimize mismatch of the clock pair in an on-chip clock generator, two DCO's are arranged symmetrically during floorplan and layout, i.e., one DCO is performed first, then the second DCO is duplicated. Last, two DCO's are placed together to complete the clock pair.

In digital designs, all logic streams are controlled by the phase of a reference clock. Ideally, each logic gate receives an identical phase of reference clock. However, both clock skew and signal racing significantly degrade overall performance. As a result, we use clock tree distribution to manage system clocks. The overall features of each module are given in Table II. Testing results show that the function of clock recovery can recover $f_{\text{clock}}/4$ NRZ data at a maximum $f_{\text{clock}} = 165$ MHz within one-data-transition acquisition with ($\tau_{\text{input}} \pm \tau_{\text{input}}/2$) pulling range. Fig. 11 shows the measured result of clock recovery at 125-MHz high-speed clock/12.5- and 9-Mbps NRZ input. The histogram of an on-chip clock generator at 166 MHz/3.3 V is shown in Fig. 12 with rmsΔ : 11.42 ps/P_kP_k : 74 ps (reference: 40 MHz $\pm$ 116 ps/output: 119 MHz with rmsΔ : 12.73 ps/P_kP_k : 88 ps). The relation between the reference and target clock is shown in Fig. 13, with reference at 33 MHz $\pm$ 108 ps and output at 166 MHz $\pm$ 37 ps ($L = 1$, $M = 40$, $N = 8$). Mismatched problem within a clock pair is less than 0.069 ns or 0.83%. Comparison data with conventional frequency synthesizers in items of acquisition time, jitter, operation frequency, power consumption, pulling, and locking range are shown in Table III. A chip microphoto of the clock recovery with the function of frequency synthesizer and on-chip clock generator is shown in Fig. 14.

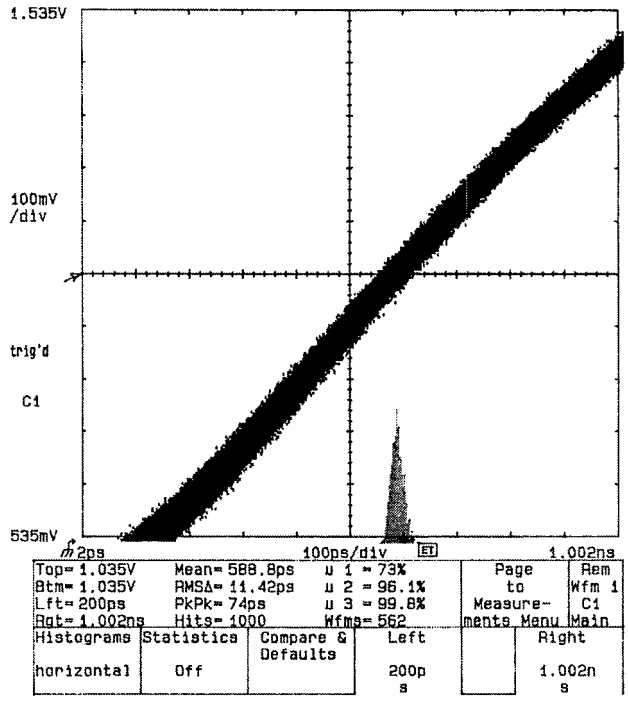


Fig. 12. Histogram of the on-chip clock generator at 166 MHz/3.3 V.

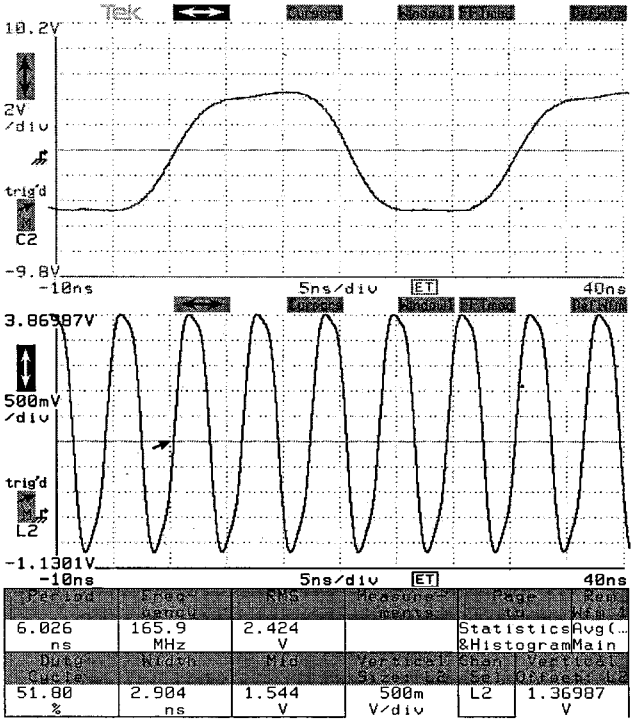


Fig. 13. Test results of the on-chip clock generator clock for $L = 1$, $M = 40$, and $N = 8$ with (a) reference clock at 33 MHz $\pm$ 108 ps and (b) output at 166 MHz $\pm$ 37 ps (mismatch $<$ 0.069 ns).

V. Conclusion

To exploit advances of digital VLSI, a new algorithm with fast acquisition and large pulling range has been presented in this paper. Based on an in-house 0.6-μm CMOS SPTM standard cell library, several test keys have been designed, fabricated, and tested. Results show that these cell-based solutions can meet many requirements in data communications. In addition, the design cycle can be greatly reduced. Also, system turnaround can be reduced during technology migration. As a result, the proposed portable and cell-based fully CMOS integrated solutions become available for communications.

TABLE III
Comparison Between Conventional Frequency Synthesizers and Our Approach

Items \ Approaches	JSSC'94 [3]	JSSC'97 [4]	Our Design
Reference Frequency	14~18MHz	NC	up to 50MHz
Maximum Frequency	80MHz	165MHz	165MHz
P_kP_k Jitter	100ps	81ps	88ps
RMS Δ Jitter	4^0-7.5^0 rms (phase noise)	13.46ps	12.73ps
PLL Bandwidth	7~26kHz	NC	Auto-Scan
Lock-in Time	NC	15us	a*log2n <14us
Programming resolution	~0.1%	linear	~0.5%
Supply Voltage Range	NC	2.5V to 7V	2.4V to 6V
Power Dissipation	30/125mW (DC/AC)	500uA (VCO)	44mW@2.4V/100MHz
Methodology	Analog	Analog	Digital

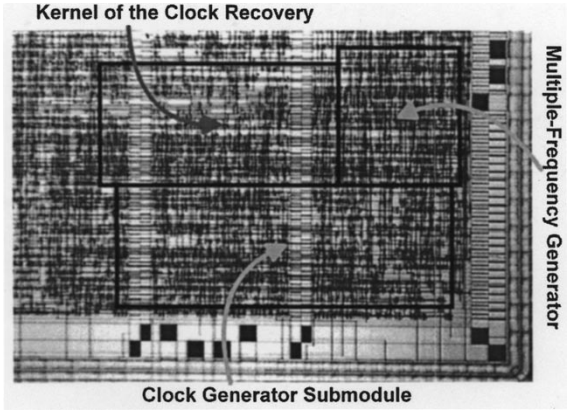


Fig. 14. Microphoto of the clock recovery and the on-chip clock generator.

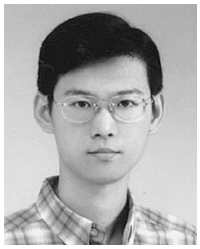
Acknowledgment

The authors would like to thank the members of the SI2 Lab of the Department of Electronics Engineering, National Chiao Tung University, for many fruitful suggestions on physical implementations. They would also like to thank the Multiple-Project Chip support from NSC/CIC.

References

[1] D. H. Wolaver, *Phase-Locked Loop Circuit Design.* Englewood Cliffs, NJ: Prentice-Hall, 1991.
[2] B. S. Song and D. C. Soo, "NRZ timing recovery technique for band-limited channels," *IEEE J. Solid-State Circuits*, vol. 32, pp. 514–521, Apr. 1997.
[3] D. Mijuskovic, M. Bayer, T. Chomicz, N. Garg, F. James, P. McEntarfer, and J. Porter, "Cell-based fully integrated CMOS frequency synthesizer," *IEEE J. Solid-State Circuits*, vol. 29, pp. 271–279, Mar. 1994.
[4] H. C. Yang, L. K. Lee, and R. S. Co, "A low jitter 0.3–165 MHz CMOS PLL frequency synthesizer for 3 V/5 V operation," *IEEE J. Solid-State Circuits*, vol. 32, pp. 582–586, Apr. 1997.
[5] H. Brugel and P. F. Driessen, "Variable bandwidth DPLL bit synchronizer with rapid acquisition implemented as a finite state machine," *IEEE Trans. Commun.*, vol. 42, pp. 2751–2759, Sept. 1994.
[6] R. E. Best, *Phase-Locked Loop Theory, Design, and Applications*, 2nd ed. New York: McGraw-Hill, 1993.
[7] A. Y. C. Wong and V. C. M. Leung, "Single tone interference rejection of code-phase multiplexed direct-sequence spread-spectrum signaling," *IEEE Trans. Commun.*, vol. 44, no. 5, pp. 557–561, May 1996.

[8] G. Brassard and P. Bratley, *Algorithmics: Theory and Practice.* Englewood Cliffs, NJ: Prentice-Hall, 1988.
[9] S.-M. Kang and Y. Leblebici, *CMOS Digital Integrated Circuits: Analysis and Design.* New York: McGraw-Hill, 1996.



Terng-Yin Hsu received the B.S. and M.S. degrees from Feng Chia University, Taichung, Taiwan, R.O.C., in 1993 and 1995, respectively, both in electrical engineering. He is currently pursuing the Ph.D. degree at the Institute of Electronics Engineering, National Chiao Tung University, Hsinchu, Taiwan, R.O.C.

His research interests include VLSI architectures, CDMA power control, wireless LAN communications, high-speed networking, system-on-chip design technology, and related ASIC designs.

Bai-Jue Shieh was born in Taipei City, Taiwan, R.O.C., on August 9, 1974. He received the B.S. degree from the Department of Electronics Engineering, National Chiao Tung University, Hsinchu, Taiwan, R.O.C., in 1996. He currently is pursuing the M.S. degree at the Institute of Electronics Engineering, National Chiao Tung University.

His specialties include IC design flow, cell-based and fully custom VLSI design, video signal processing, cell library design, and memory circuit design.

Chen-Yi Lee received the B.S. degree from National Chiao Tung University, Hsinchu, Taiwan, R.O.C., in 1982 and the M.S. and Ph.D. degrees from the Katholieke University Leuven, Belgium, in 1986 and 1990, respectively, all in electronic engineering.

From 1986 to 1990, he was with IMEC/VSDM working in the area of architecture synthesis for DSP. In 1991, he joined the Faculty of the Electronics Engineering Department, National Chiao Tung University, where he is currently a Professor. His research interests mainly include VLSI algorithms and architectures for high-speed networking, system-on-chip design technology, very-low-bit-rate coding, and multimedia signal processing.