

## Chapter 2

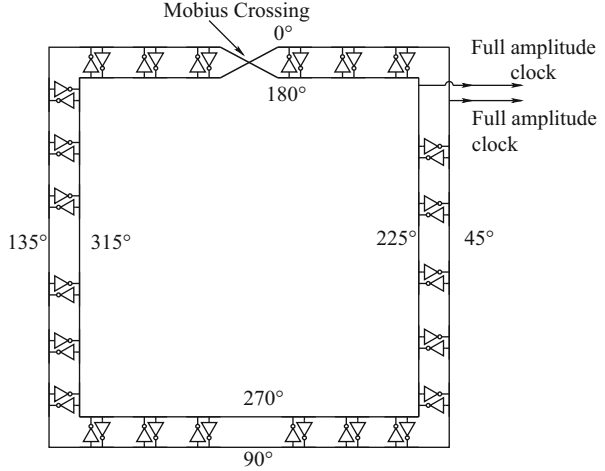
# Clock Distribution for Fast Networks-on-Chip

**Abstract** As mentioned in first Chapter, high speed, low-jitter and low-power clock distribution is a major challenge in designing a fast Network-on-Chip (NoC). In this chapter, we propose several techniques to address the issue of distributing a high-speed, low power, low jitter clock across an IC. We focus our attention primarily on resonant ring-based standing wave oscillators (SWOs) which have recently emerged as a promising technique for high-speed, low power clock generation. In Sect. 2.1, we describe the basics of resonant ring-based oscillators. In Sect. 2.2, we present a novel low-jitter, low-power clock generation and distribution methodology which uses resonant standing wave oscillators (SWOs). We also present an all-digital control loop to phase-lock the SWOs to an external reference clock. Experimental results demonstrate that the jitter of our approach is dramatically lower (by  $\sim 4.6\times$ ) than existing schemes, while the power consumption is significantly lower (by  $\sim 2\times$ ) as well. Next, in Sect. 2.3, we present a dynamic programming based approach to synthesize a minimum cost buffered H-tree for clock distribution. Our primary goal is to minimize the end-to-end jitter of the synthesized H-tree, while the secondary goal is to minimize power as well. Compared to a manually constructed buffered H-tree network, our approach is able to reduce both jitter (by as much as 28 %, and power by as much as 46 %). Finally, in Sect. 2.4, we present a tiled SWO structure with a plurality of ring-based SWO tiles arranged in a 2D fashion, oscillating at a high frequency. We validate that in this manner, an SWO approach can be used to practically implement a high-frequency, low-power clocking approach with high and uniform area coverage over an IC. Our simulations indicate that this tiled structure can oscillate at about 7.25 GHz, with low power (about 68 mW per SWO tile) and low jitter (about 3.1 % of the nominal clock period)

### 2.1 Resonant Oscillators

Recently, there has been some interest in mobius ring based resonant oscillators as a means to generate the clock signal for digital ICs. These resonant oscillators use a metallic ring along with a single (or multiple) cross coupled inverter pairs(s). Since charge is recirculated in these configurations, they exhibit a low power consumption. Resistive losses in the ring, as well as the power consumed by the inverter pair(s) contribute to the power consumption of these structures. By choosing the length of the ring carefully, oscillations of high frequencies can be sustained, as long as

**Fig. 2.1** Circuit Topology for traveling wave resonant oscillator



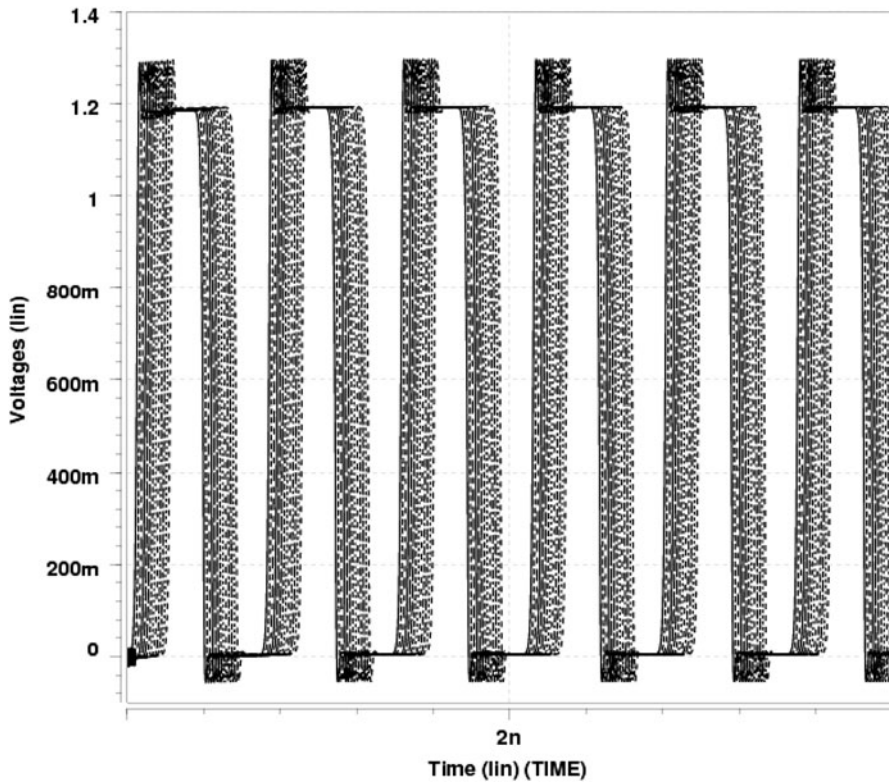
the inverter pair(s) can switch at these frequencies. Ring based resonant oscillators can be categorized into following two types: Traveling Wave Resonant Oscillators (TWOs) and Standing Wave Resonant Oscillators (SWOs).

### 2.1.1 Traveling Wave Oscillators

A *traveling wave* resonant oscillator circuit (also referred to as a *rotary clock*) was introduced in (Wood et al. 2001; 2006). The authors utilize a sufficiently long wiring ring, such that its capacitive and inductive parasitics result in a high frequency oscillatory network. This resonant clock topology is described in Fig. 2.1. Figure 2.2 shows the (overlaid) waveforms of the clock signals extracted from different locations along the ring. Oscillations in this network are sustained by a plurality of inverter pairs uniformly spaced along the ring (Fig. 2.1). However, a key drawback of the rotary clock is that the phase of the generated clock varies along the ring (as shown in Fig. 2.2), making traditional synchronous clock based design extremely difficult. Also, the clock signal at every point of the ring is a full-rail signal, resulting in a larger power consumption.

### 2.1.2 Standing Wave Oscillators

A *standing wave* resonant oscillator circuit was proposed in Cordero and Khatri 2008. In this approach, a long wiring ring is used, and oscillations are sustained in this resonant ring by just using a *single* inverter pair (Fig. 2.3). The clock signal at any point in the ring is sinusoidal, and has the *same phase* at all points along the

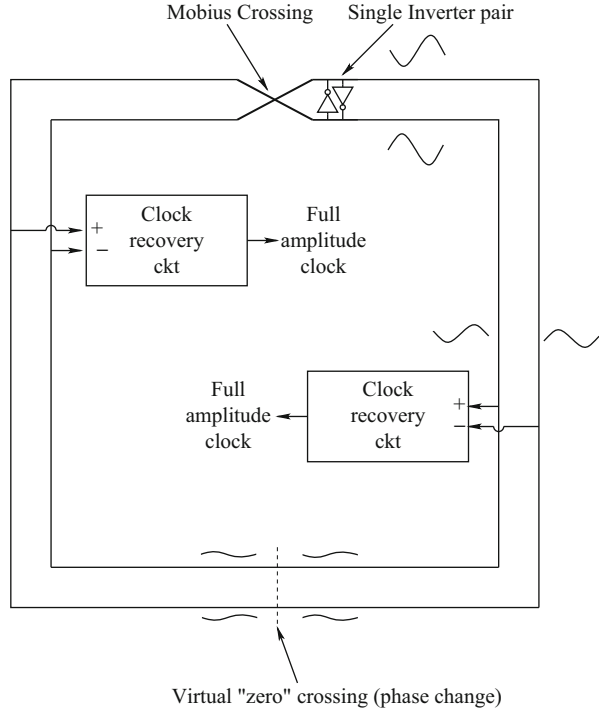


**Fig. 2.2** Sample waveforms (Overlaid) for traveling wave resonant oscillators

ring as shown in Fig. 2.4. The SWO consists of a wire ring, with a mobius crossing (shown at the top). An inverter pair is inserted at the mobius crossing location (see Fig. 2.3) to provide the negative resistance necessary to sustain oscillations. At any point in the ring, the outer wire sustains a sinusoidal oscillation of the same phase, while the inner wire sustains a sinusoidal oscillation as well, but of the opposite phase. The point diametrically opposite to the mobius crossing has a zero amplitude and is a virtual ground point (in an AC sense). A full-amplitude square wave clock is recovered at any point in the ring by using a clock recovery circuit which consists of a differential amplifier.

By using differential amplifiers at different points in the ring, full rail clock signals are extracted at the locations desired (Fig. 2.5). Thus, this approach yields full-amplitude square wave clock signals that have the same phase everywhere along the ring. This is a key improvement over the rotary clock of (Lin and Kaiser 2001), since it is compatible with synchronous IC design. In addition, the reduced ring capacitance due to the use of significantly fewer inverters (in particular, just one), increases the operating speed and reduces the power consumption as well. Note that there is an AC null (virtual “zero”) point in the center of the ring as shown in Fig. 2.3.

**Fig. 2.3** Standing wave resonant clock



As a result, the phases of the signals on the right and the left of the null point are  $180^\circ$  apart. Therefore, clock recovery circuits on the left have their connections reversed compared to recovery circuits on the right of the null point. Note that clock recovery is not performed near the null point, since the signal amplitude is very low near the null point. Both Figs. 2.4 and 2.5 were obtained using the same simulation conditions that were used in (Cordero and Khatri 2008).

## 2.2 Phase Locked Clock Generation and Distribution Using SWOs

As mentioned in the previous section, an SWO has following advantages over a TWO: (1) SWO yields clock signals with same phase everywhere along the ring, and hence can be naturally used as a synchronous clock distribution scheme; (2) SWO uses significantly fewer inverters compared to a TWO, which results in an increasing operating speed as well as reduced power consumption. Hence, we focus our attention on an SWO-based design. In the following section, we propose a phase-locked SWO-based high-speed, low power clock generation and distribution scheme.

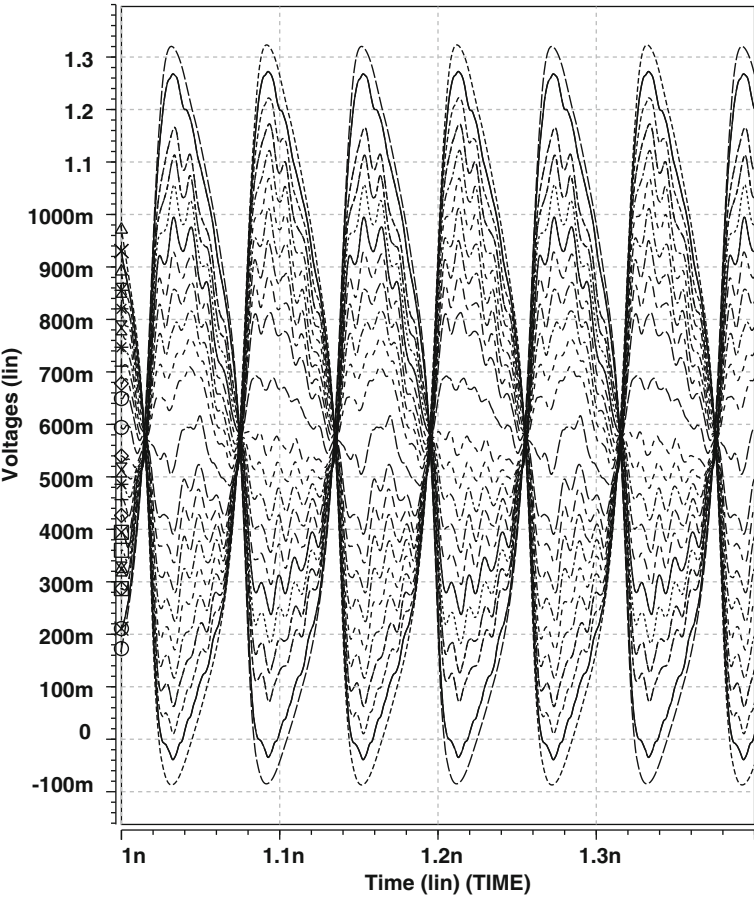
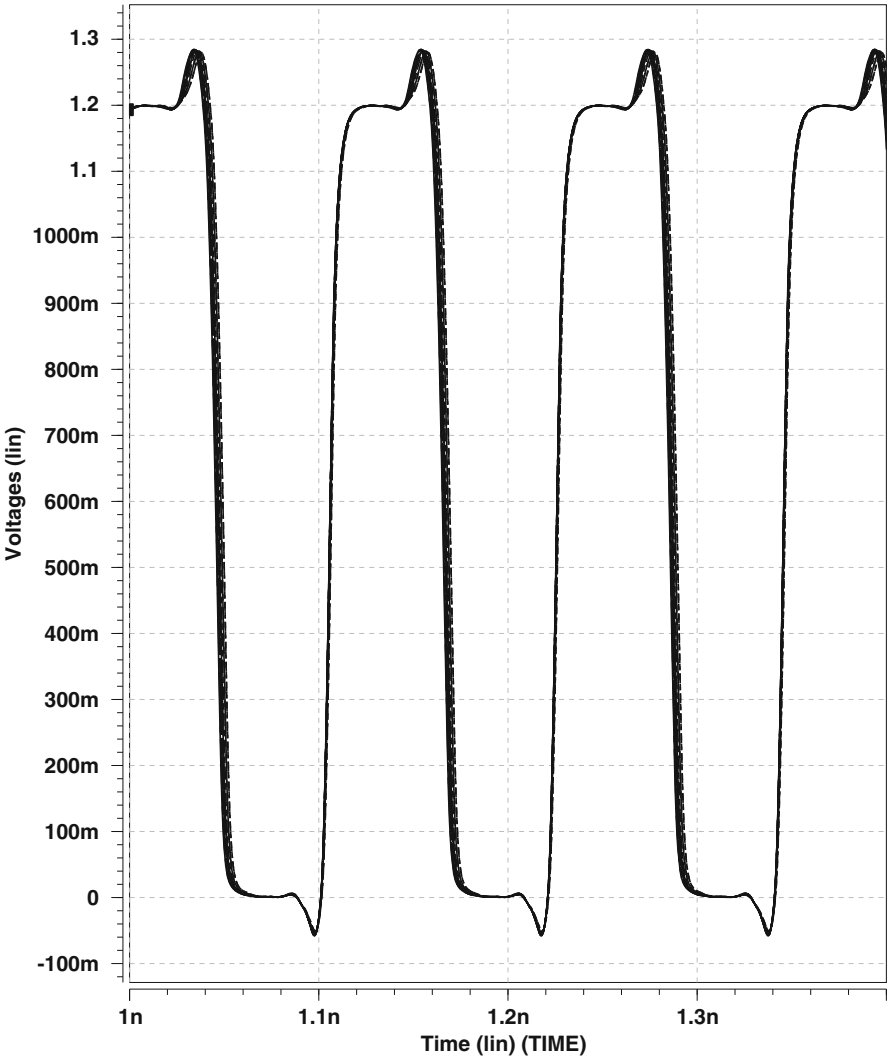


Fig. 2.4 Sample waveforms (Overlaid) for standing wave oscillator

2.2.1 Introduction

Clock generation and distribution are critically important aspects of VLSI IC design. For large digital ICs, the clock signal can contribute significantly to the power consumption of the IC. Additionally, process, voltage and temperature (PVT) variations can result in a variation in the clock arrival time at the different sinks (end-points) of the clock distribution network. This can lead to systematic skew in the clock arrival times at the different sinks, and also a dynamic variation (jitter) in the clock arrival times at the sinks due to cycle-to-cycle variations in the power supply signal at different locations in the die. A good phase-locked clock generation and distribution scheme needs to minimize the power consumption, systematic skew as well as the cycle-to-cycle jitter. Traditionally clock distribution networks have been designed using end-to-end delay of the clock signal as the key metric to minimize. We believe



**Fig. 2.5** Recovered clock waveforms (Overlaid) for standing wave oscillator

that since most ICs have an on-chip phase-locked loop (PLL), the relevant metric to minimize is instead the jitter and the systematic skew.

In this section, we present a resonant standing wave oscillator (SWO) based phase-locked clock generation and distribution scheme. Given that the scheme is resonant in nature, it exhibits a significantly lower power consumption than a traditional buffered H-tree.

Additionally, since the resonant oscillator has a relatively high Q-factor (of about 5), our scheme has superior systematic skew and jitter characteristics as well. On the

other hand, a buffered H-tree exhibits poor jitter characteristics since the buffers on any path exhibit delay variations due to local cycle-to-cycle power supply variations.

The SWO in our work is based on a ring-based oscillator shown in Fig. 2.3. Our initial experiments with the above structure indicated that the jitter characteristics of the ring-based SWO were very good. As a result, we design an all-digital control loop to vary and phase-lock the SWO frequency to an external reference, and also use the same ring topology to distribute the clock all over the IC.

Our phase-locked SWO is controlled by an all-digital control loop, which consists of coarse as well as fine frequency control. Coarse frequency control is accomplished by varying the number of oscillating wires in a bundle of wires (instead of just 2 wires as shown in Fig. 2.3). Fine frequency control is also accomplished in a digital manner, using a bank of binary-weighted capacitors connected at the inverter pairs of the ring. We present the analysis and circuit simulation results that validate the correct phase-locking behavior of the SWO-based, all-digitally controlled PLL. In addition, we utilize the same ring-based structure to distribute the clock signal all over the IC as well, in a “comb” like clock distribution topology. Our proposed clock distribution is entirely resonant in nature, and hence consumes low power. Because of the relatively high Q-factor of the SWO, jitter and systematic skew are reduced as well.

The complete SWO-based phase-locked clock generation and distribution scheme has been simulated in HSPICE (Inc Meta-Software) using a 32 nm (PTM 2013) technology. The key contributions of this work are:

- We present an all-digital control loop to phase-lock the SWO to an external clock, by combining a coarse and fine control loop over the frequency of interest (which is between  $\sim 3.1$  GHz to  $\sim 3.6$  GHz). Our phase-locking circuit does not require any external analog supply voltages for operation.
- The phase-locked SWO is also used as a clock distribution mechanism, by routing the SWO wires in a “comb” like manner to all the clock distribution end-points on the die.
- Due to the resonant nature of the clock generation and distribution scheme, a reduced power consumption (upwards of  $2\times$  lower) is achieved.
- Additionally, the jitter and systematic skew characteristics of our clock generation and distribution scheme are significantly (about  $5\times$ ) better than those of a traditional buffered H-tree scheme with an overlaid mesh.

The remainder of this section is organized as follows: We discuss some previous work for clock design in Sect. 2.2.2, while Sect. 2.2.3 provides the details of our SWO-based clock generation and distribution methodology. Section 2.2.4 presents results from experiments which we conducted, while Sect. 2.2.5 presents conclusions.

### 2.2.2 Previous Work

Recently, there has been some interest in mobius ring based resonant oscillators as a means to generate the clock signal for digital ICs. Both traveling wave (Wood et al. 2001; Chan et al. 2004; MultiGig 2013; Nedovic et al. 2007) and standing wave (Cordero and Khatri 2008; Karkala et al. 2009) oscillators have been proposed in the literature. Since charge is recirculated in these configurations, they exhibit a low power consumption. Resistive losses in the ring, as well as the power consumed by the inverter pair(s) contribute to the power consumption of these structures. By choosing the length of the ring carefully, oscillations of high frequencies can be sustained, as long as the inverter pair(s) can switch at these frequencies.

A TWO (referred to as a *rotary clock*) was described and implemented (Wood et al. 2001; 2006). The topology is similar to SWO (Cordero and Khatri 2008; Karkala et al. 2009), except that oscillations in this network are sustained by a plurality of inverter pairs spaced along the ring. The key drawback of the rotary clock is that the phase of the generated clock varies along the ring, making traditional synchronous clock based design extremely difficult. Also, the clock signal at every point of the ring is a full-rail signal, resulting in a larger power consumption. Hence, we focus our attention on a SWO-based design.

Both the traveling wave and standing wave oscillators generate a free-running clock signal. In practice, however, it is crucial that any oscillator in a digital system has the ability to modify its phase and frequency in a predictable manner, so as to allow it to be integrated into a PLL. In this section, we design a SWO-based PLL as well as a clock distribution network, with minimal power, jitter and systematic skew.

The authors of (Nedovic et al. 2007) describe a 2.5 GHz PLL using a traveling wave oscillator. The design recovers 16 bits within a clock period. Though it is advantageous for Clock Data Recovery (CDR), the varying phase of the traveling wave clock makes synchronous clock based design difficult. In contrast, our work uses an SWO-based PLL, and simultaneously accomplishes resonant clock distribution as well, with low power, jitter and skew.

In Mahony et al. (2003), a high-frequency standing wave PLL was proposed. It uses multiple coupled oscillators, each comprised of an NMOS cross-coupled pair to sustain the oscillation, and a PMOS diode connected load for setting the common mode voltage. Unlike our approach, (Mahony et al. 2003) achieves a very small (6.4 %) locking range (while we achieve a  $\sim 15$  % locking range from  $\sim 3.1$  GHz to  $\sim 3.6$  GHz). Further, the approach of (Mahony et al. 2003) does not focus on jitter, systematic skew or power, and does not concern itself with clock distribution, unlike our approach.

PLLs have been important blocks in the field of VLSI for the past few decades. Several PLLs have been designed with a coarse and a fine tuning approach. For example, the authors of (Lin and Kaiser 2001; Wilson et al. 2000; Lin and Lai 2007) have implemented a PLL using a combination of an analog and digital control loop for coarse and fine tuning. However, none of the oscillators in (Lin and Kaiser 2001;



Wilson et al. 2000, Lin and Lai 2007) were resonant. Also, in further contrast, our work uses an all-digital control, and provides resonant clock distribution as well.

Recently, in (Karkala et al. 2009), a SWO-based PLL was presented with a digitally controlled coarse frequency loop, and an analog control for the fine frequency loop. Although we borrow the coarse control approach from (Karkala et al. 2009), there are significant differences between this work and (Karkala et al. 2009). In particular, (Karkala et al. 2009) utilizes an extra 2-VDD supply for the analog control of the fine frequency, which we avoid in our all-digital control approach. Also, (Karkala et al. 2009) does not address the problem of clock distribution, while our design performs resonant clock distribution as well. Without using a resonant clock distribution, the jitter and systematic skew could be significant (if distribution was done by traditional means such as a H-tree). Additionally, unlike (Karkala et al. 2009), we compare our approach with a H-tree based clock distribution approach with a mesh overlay, and quantify the gains in power (more than  $2\times$  lower), systematic skew (about  $5.5\times$  lower) and jitter (about  $4.6\times$  lower).

In Chueh et al. (2004), the authors experimented with a 4-level H-tree, with energy-recovering flip-flops and a resonant clock distribution. The resonant clock generator was operated at 250 MHz. By varying the width and spacing of the wires, they evaluated the clock skew at different leaf nodes. Unlike our approach, they did not focus on a PLL, and rely on a small design (4-level H-tree, 4 mm die size) for their experiments. Despite this, their best case reported skew was larger than ours, even with significantly wider distribution wires that were spaced much further apart. Also, no discussion or comparison of jitter was provided in (Chueh et al. 2004).

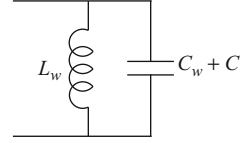
### 2.2.3 *Our Approach*

In the following section, we first briefly discuss our proposed resonant oscillator based phase-locked clock generation and distribution scheme, followed by a brief discussion of the H-tree based clock distribution scheme (with an overlaid mesh) we compare our work with. Next, we discuss the details of our all-digital PLL to phase lock the resonant clock with an external reference.

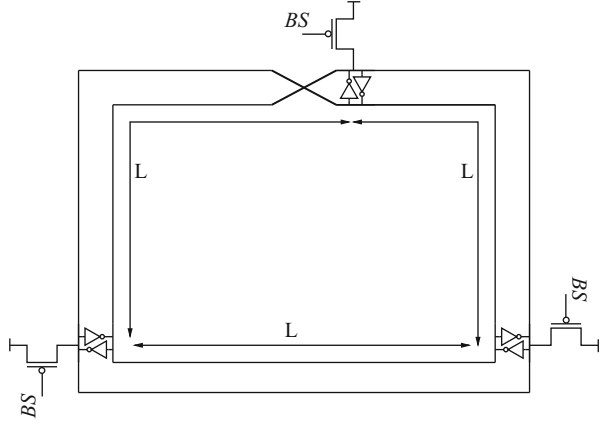
#### 2.2.3.1 SWO-based Clock Distribution

Our clock distribution scheme is based on resonant standing wave oscillators (SWOs). The physical topology of our SWO is shown in Fig. 2.3, while the equivalent circuit for our SWO is shown in Fig. 2.6. In this figure,  $L_w$  and  $C_w$  refer to the parasitic inductance and parasitic capacitance of the ring wires respectively. The capacitance due to the cross-coupled inverter pair (i.e. twice the sum of the diffusion and gate capacitances of any inverter in the pair) is  $C$ . Since  $C$  and  $C_w$  are in parallel, we obtain the equivalent circuit shown in Fig. 2.6.

**Fig. 2.6** Equivalent circuit for our resonant oscillator



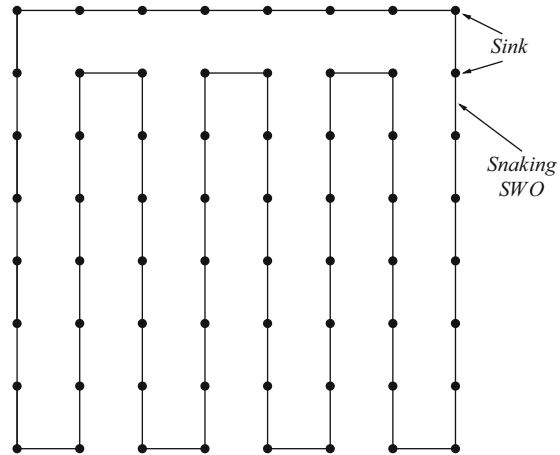
**Fig. 2.7** Longer SWO



The oscillation frequency of the equivalent circuit is given by

$$f = \frac{1}{2\pi\sqrt{L_w(C + C_w)}} \quad (2.1)$$

For reasonable oscillation frequencies, the area covered by an SWO is very small compared to the area of the typical digital IC. Hence, in order to realize a SWO based clock distribution scheme, we need to increase the size of the ring to cover a larger area of the chip. However, increasing the length of the ring decreases the oscillation frequency (due to the increase in its parasitic capacitance and inductance). Figure 2.7 shows a SWO consisting of three cross-coupled inverter pairs, placed at an equal distance from each other. The SWO of Fig. 2.7, with a length of  $3 \times L$ , oscillates at the same frequency of a single SWO of length  $L$  with one cross coupled pair of inverters. The SWO is bootstrapped with PMOS devices (using a  $BS$  signal) as shown in the Fig. 2.7 to provide the initial condition for the system to oscillate. In this manner, the SWO in Fig. 2.7 covers  $\sqrt{3}$  times the chip area as compared with a SWO with just one inverter pair, and oscillates at the same frequency. In general, we can implement an SWO with a large perimeter  $k \times L$  (where  $k$  is odd). To ensure that the clock is distributed to  $2^P$  uniformly spaced points on the die, we propose to snake the wires of the ring to implement a “comb” topology as shown in Fig. 2.8. In this figure, we assume  $P = 6$ . Since the SWO has the same phase everywhere along the ring, this “comb” structure is able to distribute the clock signal to the 64 sinks on the IC with the same phase. Note that for such a SWO design, we require an odd number  $k$  of inverter pairs, and a single mobius connection (as shown in Fig. 2.7).

**Fig. 2.8** Comb topology**Fig. 2.9** Coarse frequency configuration

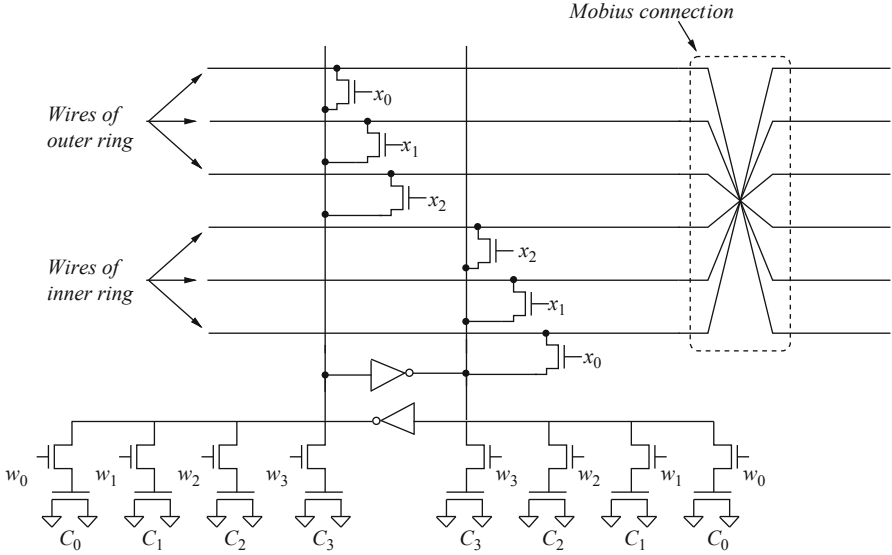
$x_0$	$x_1$	$x_2$	$x_2$	$x_1$	$x_0$	
1	1	1	1	1	1	← Coarse config 1
1	1	0	0	1	1	← Coarse config 2
1	0	0	0	0	1	← Coarse config 3
outer ring			inner ring			

### 2.2.3.2 Phase-locked SWO

We realize a phase-locked SWO by modifying the base design of SWO (Cordero and Khatri 2008). Frequency control of the SWO consists of coarse frequency control as well as fine frequency control, as discussed next.

**Coarse Frequency Control:** We implement coarse frequency control by modulating the inductance and capacitance of the ring. The two wires of the ring (Fig. 2.3) are replaced by  $2n$  parallel wires in our approach, just as in (Karkala et al. 2009). We realize a variable frequency SWO by selecting a subset (of even cardinality) of the  $2n$  wires for oscillation. The wires used for oscillation are selected symmetrically about the center of the  $2n$ -wire bundle. Each subset of the  $2n$  wires that we use for oscillation is referred to as a *coarse configuration*.

Figure 2.9 illustrates the coarse configurations for  $n = 3$ , while Fig. 2.10 illustrates the ring circuit structure for  $n = 3$ , at the mobius crossing point. In Fig. 2.9,  $x_0$  refers to the outermost wire of the outer ring as well as the innermost wire of the inner ring, while  $x_1$  refers to the middle wire, and  $x_2$  refers to the innermost wire of the outer ring, and the outermost wire of the inner ring. In this figure, a '1' against  $x_i$  indicates that the corresponding wires are oscillating, and a '0' indicates that the corresponding wires do not oscillate. Therefore, coarse configuration 1 (2) uses a total of 6 (4) wires for oscillation (indicated by the values '1' against the configuration). Note that the



**Fig. 2.10** SWO coarse and fine configuration

oscillating wires in a coarse configuration are chosen in a symmetric manner around the midpoint of the bundle of  $2n = 6$  wires. We simulated our oscillator with  $n = 30$ .

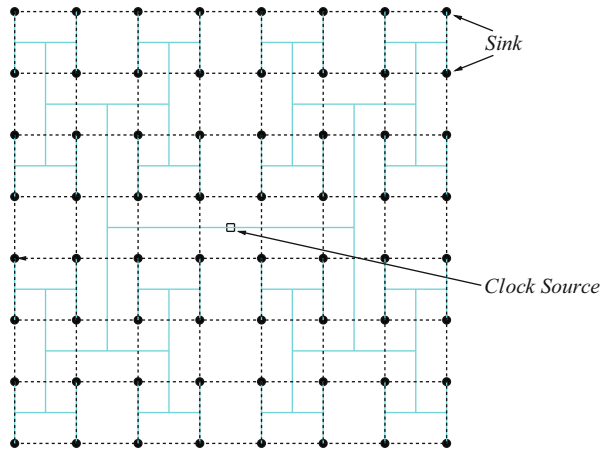
In practice, the wire locations that are labeled as '0' in Fig. 2.9 are actually left floating by the control logic. Assuming that the supply voltage of the inverter pair is  $VDD$ , the oscillating wires oscillate around a DC value  $VDD/2$ , with sinusoidal waveforms which are always in phase, but whose amplitude vary as we traverse the ring (as shown in Fig. 2.3). Since a null oscillation point exists at a distance  $L/2$  from the inverter pairs, we short all  $2n$  wires at this virtual ground location. As a result, all wires that are left floating by the control logic actually have a  $VDD/2$  voltage on them due to the short at the virtual ground location (and therefore these wires act as ground wires in an AC sense).

From Fig. 2.9, suppose we have two configurations with numerical indices  $P$  and  $Q$  respectively. Let  $P < Q$ , so that there are more oscillating wires in the inner (and outer) rings for  $P$  as compared to  $Q$ . Also, the distance between oscillating wires in the two rings is lower for  $P$ . This has two effects.

- The capacitance of the oscillating wires is larger for  $P$  as compared to  $Q$ , since  $P$  has more oscillating wires.
- The inductance of  $P$  is lower than that of  $Q$ , since the current return loop is smaller in  $P$  compared to  $Q$ , due to proximity effect.

The ratio of the increase in capacitance of  $P$  over  $Q$  is less than the ratio of the increase in inductance of  $Q$  over  $P$ . As a result, based on the expression for the frequency of oscillation of the ring (Eq. 2.1),  $P$  oscillates at a higher frequency than  $Q$ .

**Fig. 2.11** H-Tree with overlaid mesh



The coarse frequency of our resonant standing wave oscillator is controlled by varying the values of the  $n$ -bit vector  $\mathbf{x}$ .

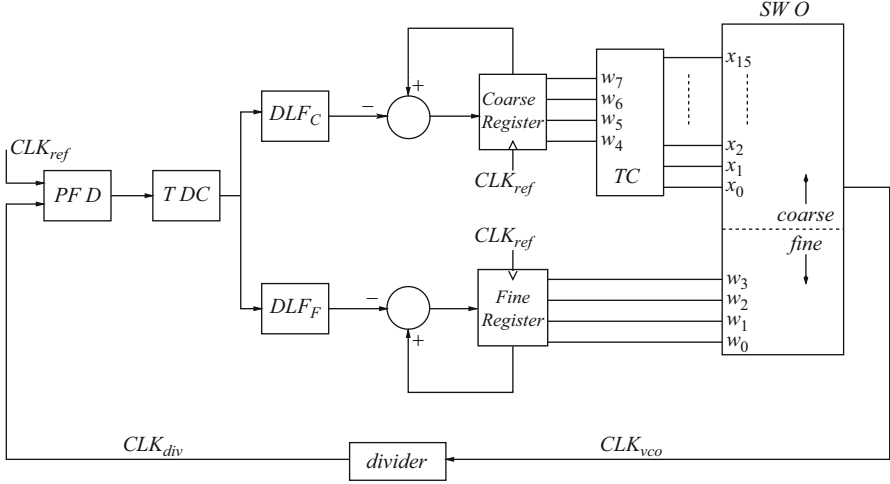
The circuitry for coarse frequency control is illustrated in Fig. 2.10. This circuit takes as an input the vector  $\mathbf{x}$ . Based on the values of this vector, the appropriate wires among the  $2n$  wires of the oscillator are made to oscillate. If coarse configuration 2 is chosen, for example, only the top 2 and the bottom 2 wires in Fig. 2.10 oscillate. Note that this circuit resides at the mobius point of the resonant oscillator, and the cross-coupled inverter pair is shown in the figure as well. The mobius connection of the  $2n$  wires is illustrated to the right of Fig. 2.10.

**Fine Frequency Control:** For fine frequency control, we use a set of binary weighted NMOS capacitors. In particular, we connect four NMOS capacitors at either end of the cross-coupled inverter pair for fine frequency control, as shown in Fig. 2.10. In practice, the switches in Fig. 2.10 are NMOS passgates. We tried complementary passgates, but the diffusion capacitance of complementary passgates caused a noticeable drop in oscillation frequency. In order to decrease the body effect, we connect the source and bulk terminals of these NMOS passgates.

The capacitor bank switches are controlled by the input vector  $\mathbf{w}$ . By applying a certain value of  $\mathbf{w}$ , we modulate the overall capacitance of the SWO and hence change its oscillating frequency. The capacitors are chosen such that they can cover the largest of the frequency intervals between any two coarse frequency points.

### 2.2.3.3 H-Tree Combined with Overlaid Mesh

We compare our approach with an H-tree with an overlaid mesh. The H-tree is a popular clock distribution approach, due to its simplicity and low power and area requirements. The structure of a 6-level H-tree clock distribution network is shown in Fig. 2.11 (with solid lines). There are  $2^6 = 64$  sinks (end-points) in the H-tree (indicated by solid dots), which are uniformly spaced to cover the entire IC.



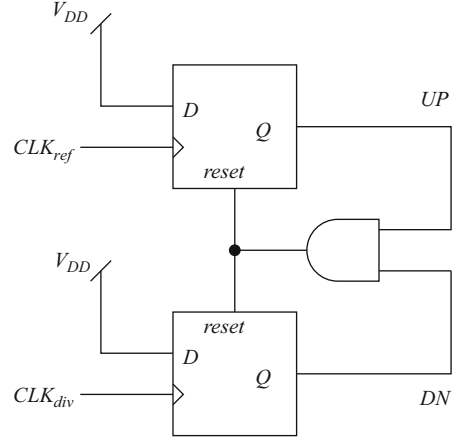
**Fig. 2.12** Block diagram of the proposed PLL design

The buffers of the H-tree (omitted in Fig. 2.11) are placed symmetrically along the branches, to ensure equal delays from the clock source to all the sinks, and fast switching at the sinks. The wire segments in a buffered H-tree exhibit minimal delay variations as a consequence of PVT variations, since the delay of a wire depends solely on the RC parasitics of the wire. However, power supply variations across the die affect the delay of the buffers, resulting in an increased skew at the sinks of the buffered H-tree. Moreover, there are cycle-to-cycle variations in the  $VDD$  value, and hence the different buffers experience different delays on a cycle-to-cycle basis. This results in an increased cycle-to-cycle jitter. Both skew and jitter reduce the maximum operating frequency of the IC. A common industry practice to reduce the skew and the jitter is to implement a mesh on top of the H-tree (shown by dotted lines in Fig. 2.11). However, by introducing short circuit current paths between the sinks with delay differences, this topology consumes more power than the H-tree alone. Moreover, the delay variation compensation of the mesh is a strict function of the RC product of the mesh wires. This implies that in order to reduce the skew and the jitter, we require a more dense mesh, with wide wires, thereby incurring a large area and power overhead.

### 2.2.3.4 Proposed Digital PLL Design

Figure 2.12 shows the block diagram of our digital PLL with the SWO of Fig. 2.10 incorporated. The phase error between the reference clock ( $CLK_{ref}$ ) and the divided clock ( $CLK_{div}$ ) is detected by the Phase-Frequency Detector (PFD) and is quantized into a digital code by the Time to Digital Converter (TDC). The TDC output is processed by the two Digital Loop Filters (DLF) for the coarse and fine frequency

**Fig. 2.13** Phase frequency detector circuit



controls. The output of the DLFs are added to coarse and fine DCO configuration words, which are registered in *Coarse Register* and *Fine Register* respectively. Note that these registers get updated every  $T_{ref}$ . A thermometer converter performs a binary to decimal encoding of the coarse control words. The output of the *TC* and the *Fine Register* are provided as input to the SWO as discussed in Sect. 2.2.3.2.

Next, we discuss the circuit details of the individual components of the PLL.

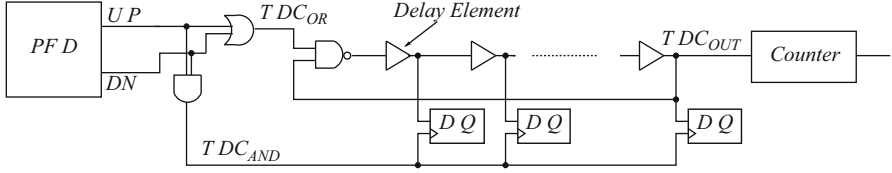
**Phase Frequency Detector (PFD):** The PFD consists of two flip-flops and an *AND* gate connected as shown in Fig. 2.13. The output of the PFD is dependent on both the phase and frequency difference between the input signals. The PFD has two input clocks, the external crystal clock ( $CLK_{ref}$ ) and the divider output ( $CLK_{div}$ ) and produces two outputs *UP* and *DN*.

The PFD is a simple state machine which has three states. Consider that the *UP* and *DN* outputs are initially low. When  $CLK_{ref}$  leads  $CLK_{div}$ , the *UP* output is asserted on the rising edge of  $CLK_{ref}$ . The *UP* signal stays high until a low to high transition of  $CLK_{div}$ . At this point of time, *DN* rises, causing both the flip-flops to reset through the asynchronous reset signal. There will be a small pulse on the *DN* output, the width of which is equal to the sum of the delay through the *AND* gate and the Reset-to-Q delay of the flip-flop. The pulse width of the *UP* signal is thus the phase error between the two signals. A similar situation arises when  $CLK_{div}$  leads  $CLK_{ref}$  (the phase error in this case is the width of *DN* pulse).

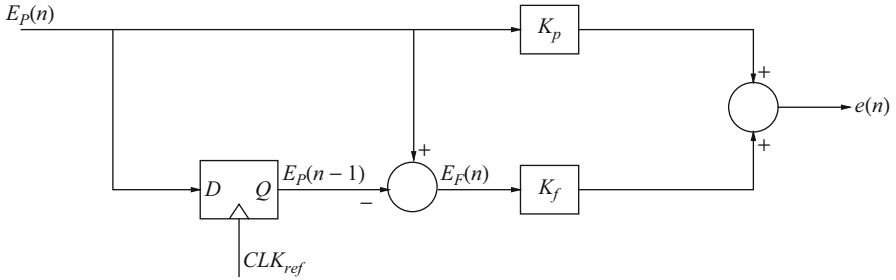
Under a lock condition, equally short pulses will be generated on both the *UP* and *DN* outputs. The transfer function of the PFD can be approximated as:

$$PFD(z) = \frac{T_{ref}}{2\pi} \quad (2.2)$$

**Time to Digital Converter (TDC):** We implement a variant of the traditional TDC with a re-circulating delay line, as shown in Fig. 2.14. It consists of  $m$  delay elements followed by a  $K$ -bit counter. Each delay element consists of 4 minimum size inverters. We define  $\Delta_{TDC}$  as the delay of a single delay element, which is the TDC resolution.



**Fig. 2.14** Time to digital converter



**Fig. 2.15** Digital loop filter

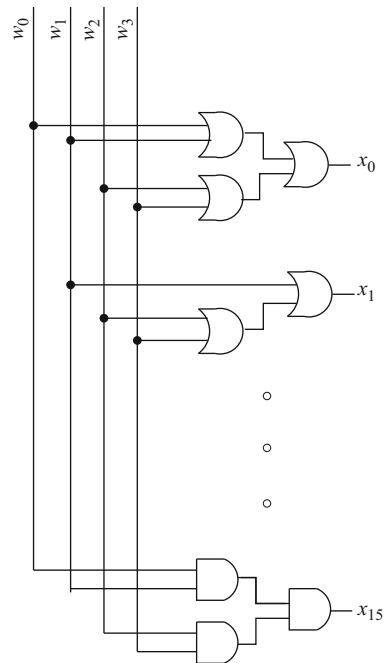
Hence,  $4 * m$  inverters and one NAND gate enabled by  $TDC_{OR}$  form an effective ring oscillator.  $TDC_{OR}$  is the logic OR, and  $TDC_{AND}$  is the logical AND of the UP and DN pulses output by the PFD. Note that, when  $TDC_{OR}$  is zero, all the outputs of the delay elements are initialized to '1'. On the rising edge of the  $TDC_{OR}$  pulse, the delay elements start toggling, causing the output  $TDC_{OUT}$  to toggle once every  $m \times T_d$  seconds, where  $T_d$  is the delay of one delay element. The counter counts the number of times  $TDC_{OUT}$  toggles. On the rising edge of the  $TDC_{AND}$ , the state of the  $m$ -element delay chain and the state of the counter are latched. Together, they perform the quantization of the phase error into a digital word. In practice, we use  $m = 5$ , and a counter with  $K = 7$  bits. The output of the TDC is  $E_P[7 : 0]$ . The 7 outputs of the counter can have 128 values. An additional 5 values (for a total of 133 values) are produced by state of the delay chain. These 133 values are encoded into  $E_P[7 : 0]$  and driven to the DLF blocks. The transfer function of the TDC can be approximated as:

$$TDC(z) = \frac{1}{\Delta_{TDC}} \quad (2.3)$$

**Digital Loop Filter (DLF)** Figure 2.15 shows our implementation of the DLF. We define two error terms called the phase error ( $E_P$ ) and the frequency error ( $E_F$ ).  $E_P$  is the measured phase error between the  $T_{ref}$  and  $T_{div}$ .  $E_F$  is the instantaneous frequency error between the  $T_{ref}$  and the  $T_{div}$ . We observe that we can approximately obtain  $E_F$  by taking a derivative of the phase error ( $E_F = \frac{dE_P}{dt}$ ). As shown in Fig. 2.15, we store the  $E_P$  of the last reference cycle in a register. We compute  $E_F$  for the current reference cycle as  $E_F(n) = E_P(n) - E_P(n-1)$ . Finally, we obtain the DLF output error



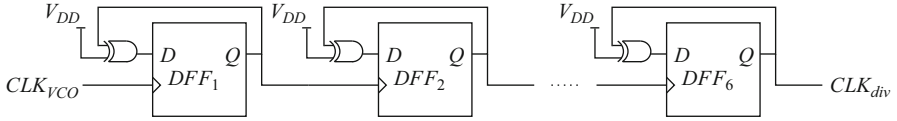
**Fig. 2.16** Thermometer converter



$e$  as  $e(n) = K_P * E_P(n) + K_F * E_F(n)$ , as shown in Fig. 2.15. Note that from control theory,  $K_P$  is defined as the proportional gain and  $K_F$  is defined as the differential gain. Hence, our DLF functions as a Proportional-Derivative (PD) controller. The proportional action improves the response of a system by providing an instantaneous response to the control error. Derivative action provides a fast response as opposed to the integral action, but cannot accommodate constant errors. PD control is useful for fast response controllers that do not need a steady-state phase error of 0. Though Proportional-Integral (PI) controllers are more common in PLL designs, we opt for a PD controller since it theoretically has an improved damping, reduced maximum overshoot as well as a very fast time to lock. Since our design implements a clock generation and distribution scheme, large overshoots result in a high cycle-to-cycle jitter. Moreover, our experiments suggest that the steady-state phase error obtained is very small and is at most  $25 \text{ ps}$ , which is less than  $0.15 \%$  of the reference clock cycle ( $T_{ref}$ ). Note that we use two different DLFs (one for the coarse, and another for the fine frequency control). The only difference between the two is that they have a different pair of gain values for  $K_P$  and  $K_F$ . The transfer function of the DLF is given by the following equation:

$$DLF(z) = K_p + K_f * (1 - z^{-1}) \quad (2.4)$$

**Thermometer Converter (TC):** Figure 2.16 shows our implementation of a thermometer converter (TC) with 4 inputs and 16 outputs, to drive the coarse configuration



**Fig. 2.17** Divider

vectors  $[x_{15} \dots x_0]$ . Lets say the decimal value of the input (vector  $w$ ) is  $m$  (where  $0 \leq m \leq 15$ ). Then exactly  $m$  outputs (from  $x_0$  to  $x_{m-1}$  are '1', and the rest of the  $(16 - m)$  outputs (from  $x_m$  to  $x_{15}$ ) are '0'.

**Divider:** In our experiments, we nominally assume an external 50 MHz reference crystal clock. Hence we require a division factor of 64 in order to achieve an oscillator operating at a center frequency of of 3.2 GHz. We implement the divider as a 6-bit ripple counter, as shown in Fig. 2.17. The clock to the first stage of the ripple counter is the recovered clock from the oscillator ( $CLK_{VCO}$ ). The output is the divided clock ( $CLK_{div}$ ) which serves an input to the phase frequency detector.

## 2.2.4 Experiments

We implemented our design in the 32 nm (PTM 2013) technology, with  $VDD = 0.9$  V. All simulations were conducted in HSPICE (Inc Meta-Software). RLC parasitics for all the wires were extracted using (Raphael Interconnect Analysis Tool: User's Guide). We assume a square die of size  $1\text{ cm} \times 1\text{ cm}$  for our experiments. We compare the jitter, skew, area and power of our approach with that of a buffered H-tree with an overlaid mesh (see Sect. 2.2.3.3).

### 2.2.4.1 Coarse Frequency Control

The inner and outer wiring rings of the oscillator consist of  $n = 30$  wires each. Each of the 60 wires were  $1\text{ }\mu\text{m}$  wide, with an inter-wire spacing of  $0.5\text{ }\mu\text{m}$ . We implement the SWO in form of a *comb* topology as shown in Fig. 2.8. For a square die of size  $1\text{ cm} \times 1\text{ cm}$ , the required length of the comb is  $80\text{ mm}$ . We implemented a  $17 \times L$  SWO, where  $L$  (the distance between two consecutive cross-coupled inverter pairs) was taken to be  $5000\text{ }\mu\text{m}$ . Note that as discussed in Sect. 2.2.3.1, the frequency of the SWO is determined by the length  $L$ , and the length of the comb should be an odd multiple of  $L$ .

The 16 coarse configurations that we used are shown in Table 2.1. Note that for each configuration, we report the 30 values of  $\mathbf{x}$ . Note that the  $x_0$  value is the leftmost bit of any row of Table 2.1. Also, the 2 highest order bits are always zero, and their corresponding NMOS devices are omitted from the circuit (see Fig. 2.10). Similarly the 2 lower order bits of  $\mathbf{x}$  are always 1, and their corresponding NMOS devices are also omitted from the circuit of Fig. 2.10. These 2 outermost wires are statically

**Table 2.1** Coarse configurations used in our experiments

Configuration	x value
0	11111111111111111111111111111100
1	1111111111111111111111111111100000
2	1111111111111111111111111110000000
3	1111111111111111111111111000000000
4	1111111111111111111111100000000000
5	1111111111111111111000000000000000
6	1111111111111110000000000000000000
7	1111111111110000000000000000000000
8	1111111111100000000000000000000000
9	1111111110000000000000000000000000
10	1111110000000000000000000000000000
11	1111100000000000000000000000000000
12	1111000000000000000000000000000000
13	1110000000000000000000000000000000
14	1100000000000000000000000000000000
15	1000000000000000000000000000000000

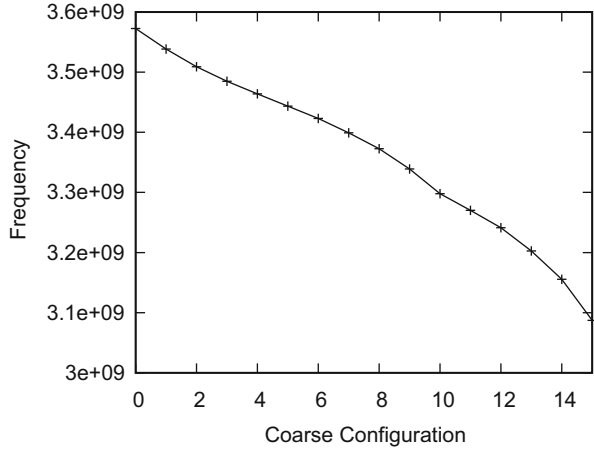
connected to the cross-coupled inverter pair. With 2 outer wires always oscillating and 2 inner wires always floating, we can choose to oscillate any number between zero through 26 of the remaining wires, giving us a total of 27 coarse configurations. However, we use only 16 of these 27 values as shown in Table 2.1. For example, while going from configuration 15 to 14, we turn on one more wire. However, while going from configuration 10 to 9, we turn on 2 extra wires. We skip some intermediate configurations because the frequency difference between some configurations and their neighbors was very small. We also ensured that we can cover the frequency difference between any two coarse configurations with the help of the fine frequency control circuitry.

We next size the cross-coupled inverters such that they can provide the negative resistance for any configuration of the SWO. Note that the cross-coupled inverters are the only active elements in the SWO. Hence, the capacitance of the cross-coupled inverter (and hence the capacitance of the SWO) depends on  $VDD$ , which can therefore result in cycle-to-cycle variations in the clock period (jitter). We found that the minimum width of the cross-coupled inverters is  $40\mu\text{m}$  ( $20\mu\text{m}$ ) for the PMOS (NMOS) device, which allowed all configurations to sustain oscillation even at a 10 % lowered  $VDD$ .

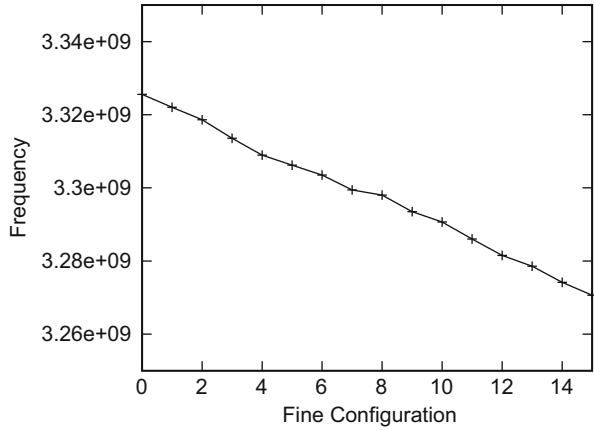
To size the NMOS passgates that drive the  $x_i$  signals in Fig. 2.10, we conducted a SPICE sweep starting with minimum sized passgates. As we increased the passgate size, we found that more configurations were able to sustain oscillation. For an NMOS passgate of size  $2.5\mu\text{m}$ , oscillations were sustained by every configuration of Table 2.1.

We verified that our oscillator provides a continuous frequency response from  $\sim 3.1$  GHz to  $\sim 3.6$  GHz, with a center frequency of 3.3 GHz. Figure 2.18 illustrates the set of frequencies achievable by our coarse tuning approach. The fine-frequency control value for these simulations is fixed at half of its maximum value, allowing

**Fig. 2.18** Frequencies achieved through coarse tuning



**Fig. 2.19** Frequency range for fine tuning of coarse configuration 9



us to minimally extend the range reported in Fig. 2.18 in both directions. We thus achieve around 15 % tuning range around a center frequency of 3.3 GHz.

### 2.2.4.2 Fine Frequency Control

We ensured that our fine frequency circuitry can cover the largest frequency difference between any two adjacent coarse configurations shown in Fig. 2.18. We use 4 binary weighted NMOS capacitors for the fine frequency control. The width of the smallest NMOS capacitor was chosen to be  $0.2u$ . The NMOS pass gates which are used to turn on these capacitors are also binary weighted. The width of the smallest NMOS pass gate is  $0.4u$ . Figure 2.19 illustrates the continuous fine frequency range achievable for one of the coarse configurations (Configuration 9).

### 2.2.4.3 Phase Locking Results

From Eqs. 2.2, 2.3 and 2.4, the open loop transfer function of the of the PLL is given by the following equation:

$$H(z) = \frac{T_{ref}}{2\pi} * \frac{1}{\Delta_{TDC}} * [K_p + K_f * (1 - z^{-1})] * K_{DCO} \quad (2.5)$$

where  $K_{DCO}$  is the DCO gain. The DLF coefficients depend on two factors  $K_{DCO}$  and  $\Delta_{TDC}$ .  $K_{DCO}$  varies from 34.2 MHz/unit to 68.3 MHz/unit for the coarse frequency, and is  $\sim 2.1$  MHz/unit for the fine frequency control. We implement the smallest delay element in our TDC with a chain of 4 minimum size inverters. Hence the delay resolution of TDC ( $\Delta_{TDC}$ ) is the sum of delay of these 4 inverters ( $\sim 25$  ps). We choose the DLF coefficients in order to (a) ensure the stability of the control loop and (b) minimize the maximum overshoot in frequency during PLL locking. For fine frequency control, we choose the phase gain ( $K_p$ ) as  $\frac{1}{2}$  and the frequency gain ( $K_f$ ) as 1. For the coarse frequency control, we choose the phase gain ( $K_p$ ) as  $\frac{1}{32}$  and frequency gain ( $K_f$ ) as  $\frac{1}{16}$ . Note that the gain of the DCO is at most  $\sim 16\times$  larger for the coarse frequency control compared to the fine frequency control. Hence, we adjust our DLF gains by the same amount. We verified the stability of the both coarse and fine frequency control by doing extensive MATLAB simulations around the values listed above.

Figure 2.20 shows the DCO frequency (top plot) and the measured phase error between the divided clock and the reference clock (bottom plot). For all our simulations, we observe that the PLL locks within 15-25  $CLK_{ref}$  cycles. With a reference crystal frequency of 50 MHz, the above locking time is only  $\sim 0.5\mu s$ . Note that we make changes in both fine and coarse frequency every  $T_{ref}$  cycles. Hence, we observe that there are flat lines in the DCO plot in Fig. 2.20, during which the configurations (and hence the DCO frequency) remain constant. We have verified that our PLL was able to correctly lock to any reference frequency (in the locking range) by testing it with different reference frequencies and different initial phase errors. Figures 2.21 and 2.22 show the simulation results of two such experiments. Observe that in both Figs. 2.20 and 2.21 we lock to the same target frequency of 3.33 GHz. However, the initial phase errors (as shown by the left portion of the two bottom plots) are different. In Fig. 2.22, the PLL locks to a different target frequency of 3.38 GHz.

To recover a rail-to-rail clock from any location on the SWO, a clock recovery circuit is implemented as shown in Fig. 2.23. This circuit is essentially a differential amplifier with a buffered output. A plot of several recovered signals from all the 17 inverter points around the SWO is shown in Fig. 2.24. From this figure, the rising slew is 5.83 ps, while the falling slew is 6.23 ps (for a clock of period of 300 ps). Note that the clock edges from all the inverter points coincide, suggesting almost zero skew (under no PVT variations).

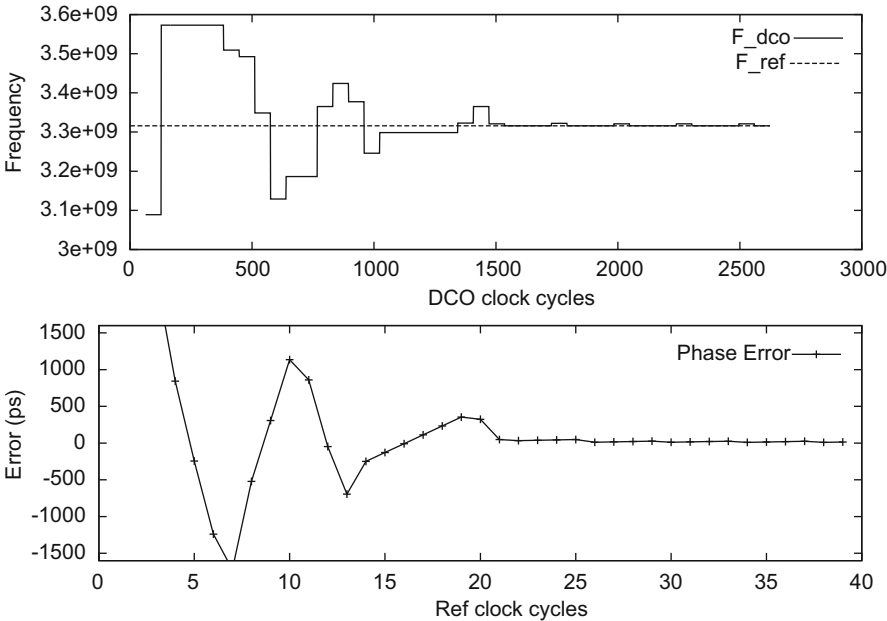


Fig. 2.20 PLL locking for a target frequency of 3.33 GHz

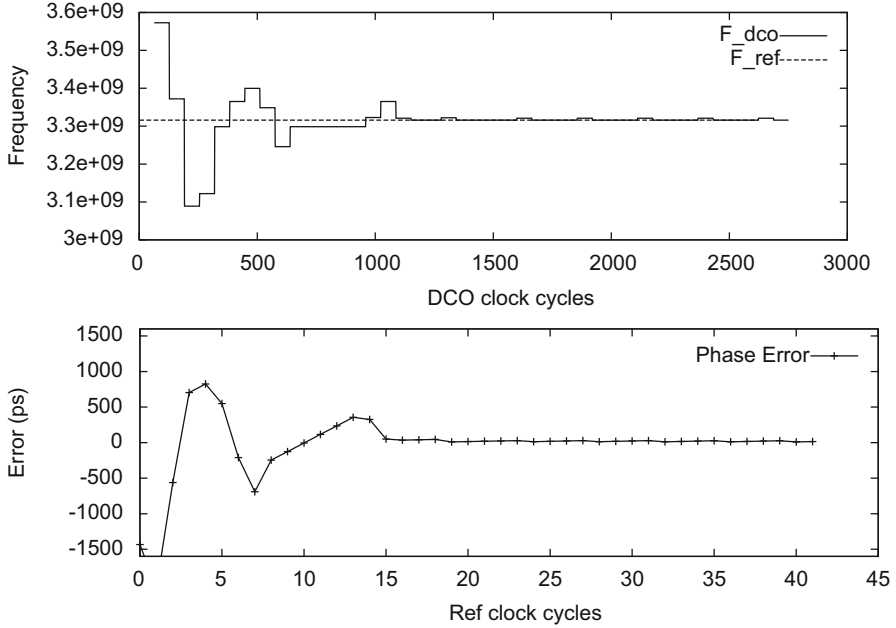
**Table 2.2** Comparison between H-tree and SWO-based clock distribution

Scheme	Wiring Area	Circuit Area	Power	Jitter	Skew
H-tree	2.48 mm <sup>2</sup>	137.85 $\mu$ m <sup>2</sup>	198.1 mW	30.5 ps	192.8 ps
SWO	8.67 mm <sup>2</sup>	141.54 $\mu$ m <sup>2</sup>	81.31 mW	6.65 ps	34.7 ps

2.2.4.4 H-tree with Overlaid Mesh—Comparison

We construct a 6-level H-tree with an overlaid mesh as shown in Fig. 2.11. The width of the clock wire for the H-tree was taken to be  $0.45u$ . The mesh was implemented with wires of thickness  $10u$ . Our clock buffer consists of a  $85\times$  inverter followed by a  $256\times$  inverter. We insert two buffers at every bifurcation of the H-tree, one for each branch. We also make sure that the longest wire that we drive is less than 1.25 mm, to ensure that the slew is less than 50 ps at the buffer inputs.

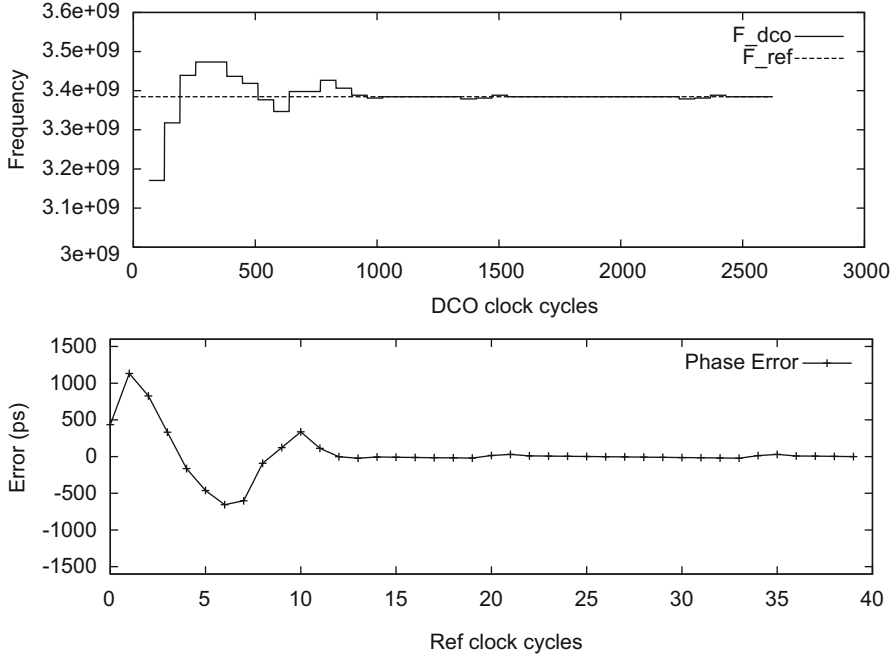
Table 2.2 compares our SWO clock distribution with the H-tree combined with an overlaid mesh. We assume 60 mV of  $V_{DD}$  variation from cycle-to-cycle in our experiments to calculate cycle-to-cycle jitter (variation in clock period between consecutive clock cycles). The  $V_{DD}$  variation of 60 mV was obtained based on similar data measured from contemporary ICs (Juniper Networks 2010). To calculate the skew (variation of clock edges across different clock sinks on the die), we divide the die into 4 quadrants. For the North-East quadrant, we assume all the transistors to be in slow corner (10 % lower  $V_{DD}$ , 10 % higher  $V_t$ , 80°C operating temperature). The North-West and South-East quadrants have nominal  $V_{DD}$  and  $V_t$ , and operate at



**Fig. 2.21** PLL locking with a different initial phase error

room temperature. For the South-West quadrant, we assume all the transistors to be in fast corner (10 % higher  $VDD$ , 10 % lower  $V_t$ , 0°C operating temperature). Our experimental conditions are pessimistic in order to compare the performance under worst case variations.

From Table 2.2, we note that our scheme has a much lowered power requirement (by  $\sim 2.4\times$ ) over the H-tree with overlaid mesh. In practice the power improvement would be larger, since we do not model the PLL power for the H-tree with overlaid mesh (while this portion of the power is accounted for in our method). This reduced power is attributed to the resonant nature of the SWO. The jitter of our approach is a mere 6.65 ps, which is  $4.6\times$  lower than that of a buffered H-tree with an overlaid mesh. Again, this attributed to the relatively high Q-factor (which is approximately 5) for our SWO. Also, the skew of our approach is about  $5.5\times$  better than the buffered H-tree with an overlaid mesh, for the same reason that results in improved jitter. The circuit area of the two approaches are similar, while the wiring area of our approach is  $\sim 3.5\times$  higher. Clearly for high-performance processors, where clock skew, jitter and power are significant concerns, our SWO provides a significantly improved clock generation and distribution scheme.

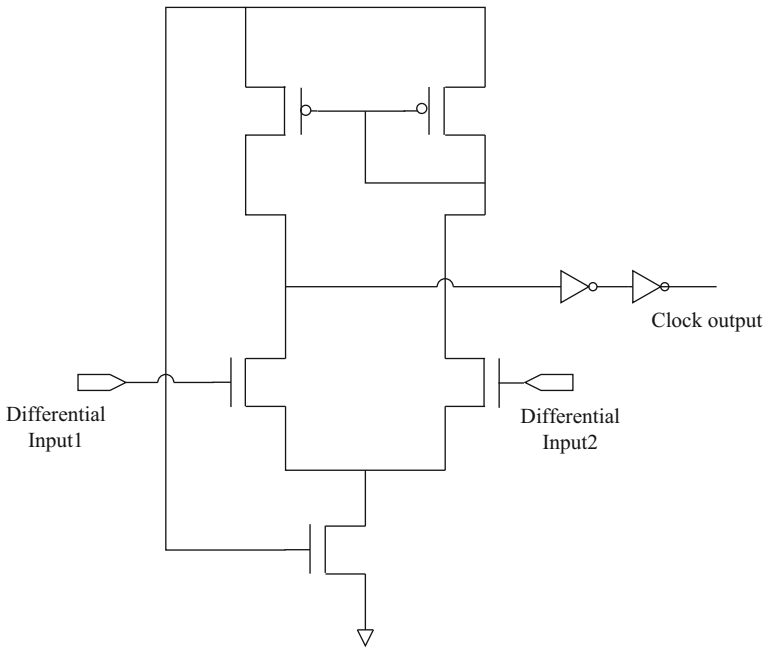


**Fig. 2.22** PLL Locking for a Target Frequency of 3.38 GHz

### 2.2.5 Conclusion

In this work, we have presented a resonant SWO based phase-locked clock generation and distribution scheme with superior jitter, skew and power characteristics. Phase locking is accomplished by a fully digital control loop consisting of a coarse frequency control and a fine frequency control. The SWO frequency is modulated in a coarse manner by changing the ring inductance, by modifying the number of oscillating wires. Fine frequency control is achieved by modifying ring capacitance via a binary-weighted capacitor bank. The digital control loop has been analyzed and implemented in HSPICE. Clock distribution is performed by routing the resonant ring on the die in a “comb” manner. Our approach is compared to an H-tree with an overlaid mesh. Given the relatively high Q-factor of the SWO, this clocking approach exhibits a very low cycle-to-cycle jitter (about  $4.6\times$  better) and very low skew (about  $5.5\times$  better) compared to the H-tree with overlaid mesh. The power consumption of our scheme is upwards of  $2.4\times$  improved as well, given the resonant nature of the approach.





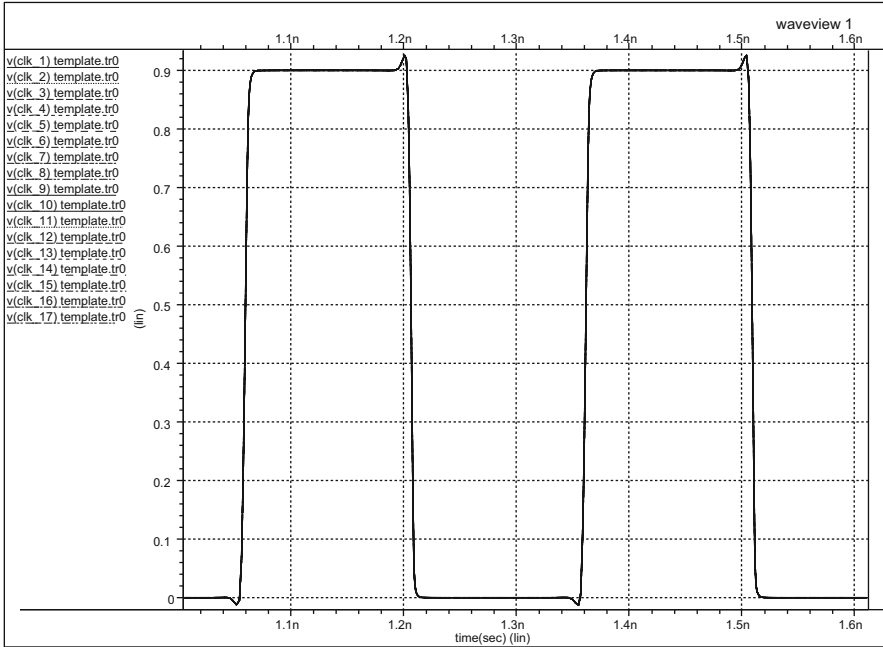
**Fig. 2.23** Clock recovery circuit

## 2.3 Automated Methodology to Generate Low Jitter Buffered H-tree

In the previous section, we presented an SWO-based clock generation and distribution methodology and compared it with a traditional buffered H-tree, with an overlaid mesh. However, in recent fabrication technologies, the buffered H-tree clock distribution network has been quite commonly used. In the following section, we present a dynamic programming based approach to synthesize a minimum cost buffered H-tree clock distribution network. Our primary cost function is the end-to-end jitter of the synthesized H-tree, while the secondary goal is to minimize power as well.

### 2.3.1 Introduction

In recent VLSI fabrication processes, with decreasing feature sizes, wire widths have been shrinking. The direct consequence of this is an increase in wire resistance ( $R_w$ ), since  $R_w = \frac{\rho \cdot L}{W \cdot T}$ , where  $\rho$  is the resistivity of the wire material, and  $L$ ,  $W$  and  $T$  are the length, width and thickness of the wire respectively. Since wiring delays on a die are proportional to the  $RC$  time constant of the wires, this increase in wire resistance results in increased wiring delays, especially for long wires on a chip

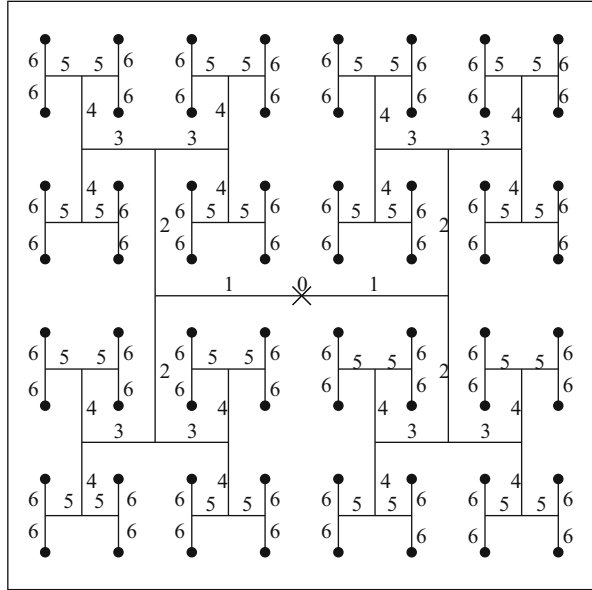


**Fig. 2.24** Overlay of recovered waveforms

(which do not scale with technology). The classical approach to solve this problem is to partially compensate for the reduction of  $W$  by increasing  $T$ , but this results in wires which are thin and tall, leading to an increased cross-coupling capacitance among wires, thereby again resulting in an increased  $RC$  time constant. The design of the wiring stack (i.e. the height and minimum width of wires on each metal layer of the IC) is therefore a complex problem. Despite significant research in academia as well as industry, it remains true that wiring delays in VLSI are on the rise (The International Technology Roadmap for Semiconductors 2007), and this problem is particularly acute for global signals which need to be driven at high frequencies with tight slew-rate constraints.

The global clock signals on an IC are particularly impacted by this problem. The clock signal is operated at high frequencies, and in addition to having stringent slew-rate constraints, the clock signal also has very tight jitter constraints. Consider the rising (falling) edge of the clock. Over a number of cycles, this edge may appear early or late. Jitter is difference between the latest arrival time and the earliest arrival time of a clock signal edge. The clock period of the IC is determined by adding the worst-case clock jitter value to the longest delay between any two sequentially adjacent flip-flops in the circuit. The longest delay between two sequentially adjacent flip-flops is computed as the sum of the longest combinational delay of the circuit plus the setup time and the clock-to-output delay of the flip-flops of the design. As

**Fig. 2.25** Buffered H-tree  
Clock Distribution Network  
with 6 Levels



a result, a high clock jitter results in a reduced frequency of operation of the IC, and hence it is important to keep jitter to a minimum.

Given the timing critical nature of the clock signal, it is very important to distribute this signal carefully on an IC, to ensure high slew-rates, high frequency and low jitter. Over the years, several clock distribution methodologies have been developed, such as the H-Tree, mesh and star, among others (Friedman 2001; Anceau 1982). The H-tree is a popular clock distribution approach, due to its simplicity and low power and area requirements. In this work, we therefore focus our attention on an H-tree based clock distribution network.

The structure of a 6-level H-tree clock distribution network is shown in Fig. 2.25. The numbers on each of the branches of the H-tree indicate the level of that branch. The  $\times$  notation in the center indicates the source of the H-tree. There are  $2^6 = 64$  sinks (end-points) in the H-tree, indicated by dots. Note that the end-points of the H-tree distribution network are uniformly spaced, and cover the entire IC area. The lengths of the  $i^{th}$  branch of the H-tree is given by  $L_i = \frac{S}{2^{\lceil \frac{i}{2} \rceil + 1}}$ , where  $S$  is the dimension of any side of the (square) die. For example, for a chip which is 20 mm wide, the lengths of the branches of a 6-level H-tree are 5 mm, 5 mm, 2.5 mm, 2.5 mm, 1.25 mm and 1.25 mm, for levels 1 through 6 respectively.

The unbuffered H-tree network is a zero-skew balanced network (if process and temperature variations are not considered). A traditional H-tree is designed assuming that the clock driver at the center of the H-tree is large enough to drive the entire clock tree. Wire widths and driver sizes are fixed to make sure that the clock signal can drive the local clock regenerators at the leaves (sinks) of the H-tree for the required frequency of operation, with a sufficiently high slew-rate. The optimal (with respect

to having a high slew-rate clock signal and also in terms of reducing the clock distribution area and delay) wire sizing methodology dictates that we utilize wider wires near the center of the H-tree, and narrower wires as we get closer to the leaves. This is referred to as a *tapered* wire sizing approach.

Note that with increasing wiring delays in recent technologies, it is no longer feasible to utilize unbuffered H-trees for ICs of reasonable size (greater than  $\sim 5$  mm). For this reason, buffered H-trees are commonly utilized in today's IC designs. The buffers of the H-tree are not shown in Fig. 2.25, and can be at arbitrary locations along any path from the source to a sink, provided the locations and sizes of each buffers is the same for all 64 paths of the H-tree. Typically, when an H-tree is manually designed (which is the common practice), designers typically place identical buffers at regular intervals along the H-tree, so as to simplify the design process. Also, in a manually designed H-tree the wire topologies are typically fixed from source to all sinks. For example, a common practice is to place buffers at each branching site in the H-tree. Note that with a buffered H-tree, tapered wire sizing (commonly used in unbuffered H-trees) is no longer necessary, and the wire widths of each level of the H-tree are typically dependent on the size of the driving buffer.

The buffered H-tree alleviates the wiring delay problem in recent technologies, and enables a designer to deliver a high frequency, high slew-rate clock at all the sinks, with low end-to-end delay. However, the disadvantage of such a buffered H-tree is an increase in the jitter at the sinks. In practice, there are cycle-to-cycle variations in the  $VDD$  value across the die. On one hand, the wire segments in a buffered H-tree do not exhibit delay variations as a consequence of the cycle-to-cycle  $VDD$  variations along the die, since the delay of a wire depends solely on the RC parasitics of the wire. However, the cycle-to-cycle variations in the  $VDD$  value across the die affect the delay of the buffers in a buffered H-tree, since the delay of an inverter depends on its supply voltage. Hence, in practice, the different buffers in a buffered H-tree experience different delays on a cycle-to-cycle basis. This results in an increased jitter at the sinks of the buffered H-tree. This can reduce the maximum operating frequency of the IC, since the operating frequency has to be guard-banded to account for worst-case jitter, as discussed earlier.

Typically a buffered H-tree network is designed to minimize the end-to-end delay of the tree. However, most modern ICs have an on-chip Phase Locked Loop (PLL) to synchronize the off-chip clock (typically generated by a crystal oscillator) to the on-chip clock. As a consequence, the end-to-end delay of the H-tree (or any other clock distribution network) is not of consequence. Rather, our position is that the H-tree should be designed for minimum jitter. This is the key guiding principle of this work. We show that designing a buffered H-tree for minimum end-to-end delay can yield different results compared to designing the buffered H-tree for minimum jitter.

Our approach automates the design of the buffered H-tree, thereby availing significant optimization flexibility that is not possible to leverage in a manual H-tree design process. In particular, we select the best H-tree by exploring various (a) wiring configurations (referred to as *wire-codes* in the sequel), (b) buffer sizes (c) segment

lengths and (d) segment topologies. Any given wiring segment in our approach has a total of 2745 choices for its implementation.

The automatic H-tree synthesis procedure utilized in this work is based on dynamic programming (DP). First, we statically characterize a large number of ways of implementing a segment, by computing their power, segment delay and segment jitter and output slew, as a function of input slew. This is done for all 2745 segment choices up-front. Then a dynamic programming engine selects the lowest cost H-tree implementation that satisfies constraints such as output slew limits and cycle-to-cycle jitter. Our approach is implemented to minimize one of two cost functions—a weighted sum of power and jitter, and a weighted sum of power and delay. The resulting H-trees are simulated in HSPICE, and the end-to-end delay, power and end-to-end jitter estimated by our DP engine matches the HSPICE results with a maximum error of 4.6 %. We also compare our DP based buffered H-tree synthesis with a manually designed H-tree. Results demonstrate that our approach, when run with jitter minimization as its goal, produces H-trees with strictly better jitter (by as much as 28 %) and power (by as much as 46 %) compared to a manually designed H-tree. Also, it achieves better jitter performance than the DP based approach run with delay minimization as its goal.

The key contributions of this work are:

- We argue that buffered H-trees should be designed to minimize jitter (as opposed to minimizing end-to-end delay which is the typical approach). We demonstrate that synthesizing an H-tree for minimal end-to-end delay does not necessarily minimize jitter, underscoring the importance of the above observation.
- We present a DP based automated approach to synthesize a buffered H-tree. This approach simultaneously explores a large number of wiring wire-codes, buffer sizes, segment lengths and segment topologies.
- With the cost function of minimum jitter, our approach produces an H-tree which has up to 28 % lower jitter than a manually designed H-tree. The power as well as the end-to-end delay of our approach is better by up to 46 % and 16.6 % respectively.
- Our DP engine can produce H-trees with a cost functions of a weighted sum of power and jitter, and a weighted sum of power and delay.

The rest of this section is organized as follows: Sect. 2.3.2 describes previous approaches in this area. Section 2.3.3 describes our approach, while Sect. 2.3.4 describes the results of experiments which we performed to validate our approach. In Sect. 2.3.5, we draw conclusions.

### 2.3.2 Previous Work

Several clock distribution methods (such as the H-trees, meshes, stars etc.) have been studied in the past. Some excellent survey style papers in this area are (Friedman 2001; Anceau 1982). Most of previous works on clock network design

(Chen and Wong 1996; Liu et al. 2000; Tsai et al. Tsai et al. 2004) attempt to obtain zero skew, while others try to bound the clock skew within a given hard constraint (Cong et al. 1998). Some clock synthesis works generate unbuffered clock networks that require a separate buffer insertion stage using conventional or specialized buffer insertion algorithms. In other works (Rajaram and Pan 2006; Chaturvedi and Hu 2004), buffer insertion and clock tree routing are integrated. The Deferred-Merge Embedding (DME) algorithm (Chao 1992) is the fundamental technique used in many recent clock tree synthesis approaches. It consists of a bottom-up stage and a top-down stage. A tree of merge segments, which represent the possible locations of merge nodes, is constructed in the bottom-up stage. Once all the merge segments are constructed, the exact locations of merge nodes are determined in the top-down stage. Merge segments are calculated to balance the delays of the two sub-trees. A key difference between DME and our approach is that DME is used for clock tree synthesis for ASICs, unlike the H-tree approach of our work which is more pertinent for custom IC designs. In this work we intend to develop an optimal H-Tree that will minimize the appropriate cost function by exploring a large number of wiring wire-codes, buffer sizes, and segment length and segment topologies. This automatic tool gives the flexibility to the designer to optimize delay, power and/or jitter across the clock tree. None of the previous approaches attempts to minimize jitter as a design goal.

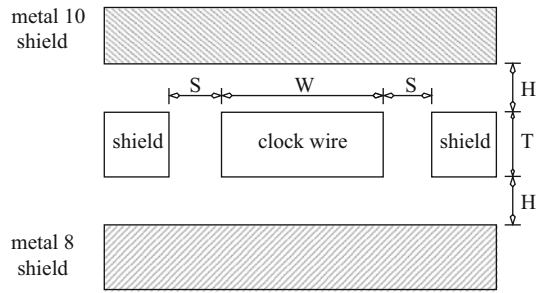
### 2.3.3 *Our Approach*

Traditionally, buffered H-trees have been designed manually, with the goal of minimizing end-to-end delay. Since the end-to-end delay is unimportant in an IC which has a PLL, and since the buffers in the H-tree introduce jitter at the clock sinks, the goal of this work is to automatically synthesize a buffered H-tree, which minimizes the end-to-end jitter of the synthesized H-tree. The secondary goal is to minimize power as well.

In the remainder of this section, we will discuss the design scenario for this effort, along with the electrical constraints which we impose on the resulting synthesized buffered H-tree. We end with a discussion of our dynamic programming (DP) based automated buffered H-tree synthesis approach.

#### 2.3.3.1 Buffered H-tree Construction

We conduct our experimentation on a 6 level buffered H-tree, implemented in a 45 nm (PTM 2013) fabrication process. The H-tree is implemented on METAL9, and we assume that 10 metal layers are available. A variety of buffers are used. Each buffer consists of two inverters, with the driving inverters of size  $32\times$  to  $51264\times$  of a minimum sized inverter, in increments of  $32\times$ . The other inverter in the buffer is sized  $3\times$  smaller than the driving inverter.

**Fig. 2.26** Cross section of wires in any H-tree segment**Table 2.3** Wire-codes and their characteristics

Wire-code	$W(\mu)$	$S(\mu)$	Res. ( $\Omega/\text{mm}$ )	Cap. (F/mm)
WC1	0.45	0.45	69.78	1.68E-13
WC2	0.45	0.9	69.78	1.08E-13
WC3	0.9	0.45	34.89	1.84E-13
WC4	0.9	0.9	34.89	1.23E-13
WC5	1.35	0.45	23.27	1.98E-13
WC6	1.35	0.9	23.27	1.38E-13

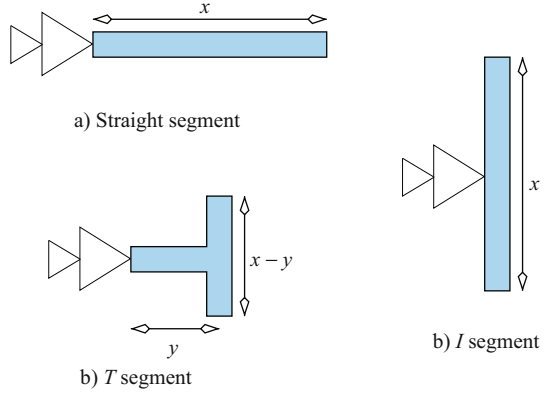
A total of 6 wiring configurations (referred to as wire-codes) are available for any buffered segment of the H-tree. There is no restriction on the number of wire-codes utilized for the synthesized H-tree. These wire-codes are referred to as WC1 through WC6, and their details are described next. Each wire-code consists of a METAL9 wire, shielded on either side as well as above and below by grounded wires, as shown in Fig. 2.26.

Table 2.3 describes the details of the 6 wire-codes, along with their parasitic resistance and capacitance values, which are extracted using Raphael (Raphael Interconnect Analysis Tool: User's Guide). All resistive values are reported for an operating temperature of  $100^\circ\text{C}$ , to model the worst case wiring delay condition. For all wire-codes, we assumed  $T = 0.9 \mu\text{m}$ ,  $H = 1.4 \mu\text{m}$  and a dielectric constant  $\kappa = 2.5$ . The shield wire has a width of  $0.45 \mu\text{m}$  in all wire-codes.

A variety of segments are selectable during the automatic synthesis of the buffered H-tree using our approach. These segments fall into three categories, as shown in Fig. 2.27. These segments are (a) a straight segment (b) an  $I$  segment and (c) a  $T$  segment. The total length of these segments is referred to as  $x$ . The straight portion of a  $T$  segment has a length  $y$ . For all segments,  $x \in [1 \text{ mm}, 2.5 \text{ mm}]$ , with increments of  $0.25 \text{ mm}$ . For a  $T$  segment,  $y \in [0.25, x - 0.5 \text{ mm}]$ , with increments of  $0.25 \text{ mm}$ .

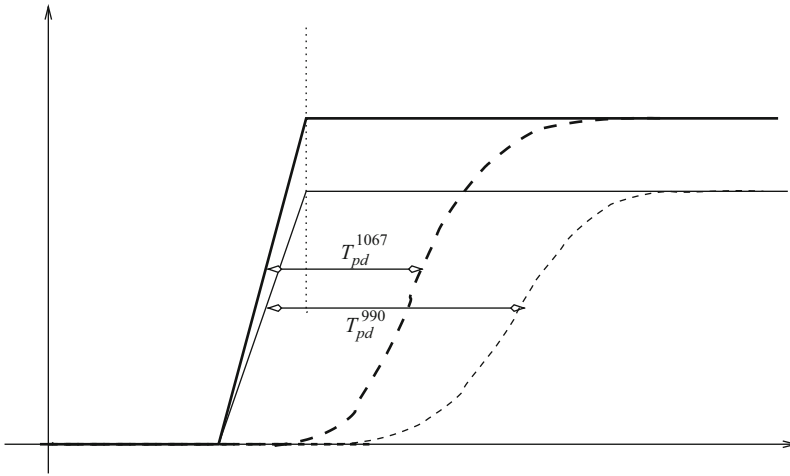
As discussed earlier, jitter in a buffered H-tree occurs due to cycle-to-cycle local variations of  $VDD$  across the die, which result in relative changes in the clock buffer delays. We next discuss our methodology for computing jitter. For our 45 nm process, the supply voltage is 1.1 V. The maximum variation in the  $VDD$  value is assumed to be 10 % of  $VDD$ , or 110 mV. This variation is due to various sources, including variations in the voltage regulator output, IR drops in the power distribution network, etc. Of this 110 mV, about 70 % is attributed to cycle-to-cycle variations, and 30 % constitutes long term variations. Therefore the cycle-to-cycle variation of the supply

**Fig. 2.27** Types of wire segments used For Buffered H-Tree construction



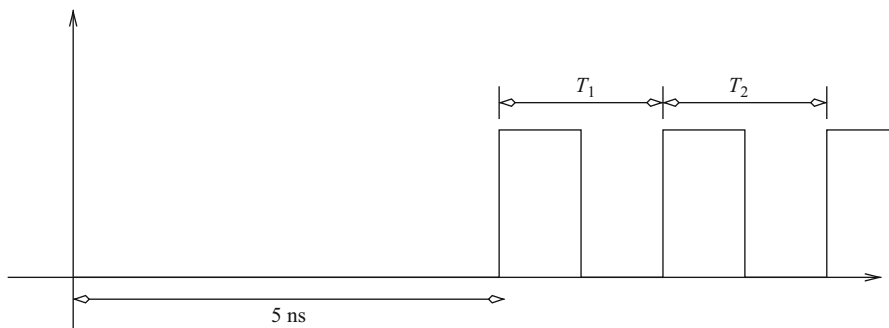
voltage is 77 mV. For purposes of jitter measurement, we therefore consider jitter to be the difference in delay of the clock buffer for  $V_{DD}$  values of 990 mV and 1067 mV (i.e. 990 mV + 77 mV). These assumptions are supported by experimental data obtained from an industrial IC design (Juniper Networks 2010).

The jitter measurement methodology is illustrated in Fig. 2.28. Consider the clock buffer operating at 1067 mV. Assume that the solid thick line is the input to the buffer, and the dashed thick line is its output. Therefore its propagation delay at 1067 mV is  $T_{pd}^{1067}$  as marked. Now consider the clock buffer operating at 990 mV. Assume that the solid thin line is the input to the buffer, and the dashed thin line is its output. Therefore its propagation delay at 990 mV is  $T_{pd}^{990}$  as marked. The jitter of the clock buffer is therefore  $J = T_{pd}^{990} - T_{pd}^{1067}$ .



**Fig. 2.28** Measuring Jitter





**Fig. 2.29** Wake-up Jitter experiment

Note that the input to the clock buffer operating at 1067 mV and 990 mV in our experiments could be driven in one of two ways—with the same slew-rate (Method A) or with the same transition time from rail to rail (Method B). Figure 2.28 shows the input of the clock buffer being driven using Method B. We use Method B instead of Method A based on experiments in which we drove a long chain of identical H-tree segments with inputs of both kinds. We noticed that the output after the signals traverse 5 H-tree segments closely match Method B, which is why we utilize this method to drive the clock buffer inputs when measuring jitter.

We additionally invoke several constraints on the wire segments utilized by our buffered H-tree synthesis algorithm

- First, if any segment being considered by the synthesis algorithm has a output slew greater than 150 ps, we reject that segment. The output slew is computed as the 10 % to 90 % (or 90 % to 10 %) transition time for the output of the segment, divided by 0.8. This constraint ensures that the clock signal at the sinks has a high slew-rate which is essential for a clock signal, and also results in reduced crow-bar currents and hence lower power.
- We also ensure the cycle-to-cycle (“wake-up”) jitter is less than 1 ps. Figure 2.29 illustrates this idea. Modern ICs frequently need to be designed in a manner in which the IC needs to be put in “sleep”, in order to save power. In such a case, the clock signal is held static during the “sleep” mode. After the IC “wakes” up, the pulse widths of the first two clock pulses must be tightly controlled for correct functionality. The wake-up jitter is defined as  $|T_2 - T_1|$  (see Fig. 2.29). We constrain each segment to have a wake-up jitter of less than 1 ps. All segments which fail this requirement are rejected. This requirement essentially requires that each segment be able to drive its clock signal to rail values during normal operation of the IC. If this is not true for any segment, such a segment will fail the wake-up jitter test.

### 2.3.3.2 Buffered H-tree Construction

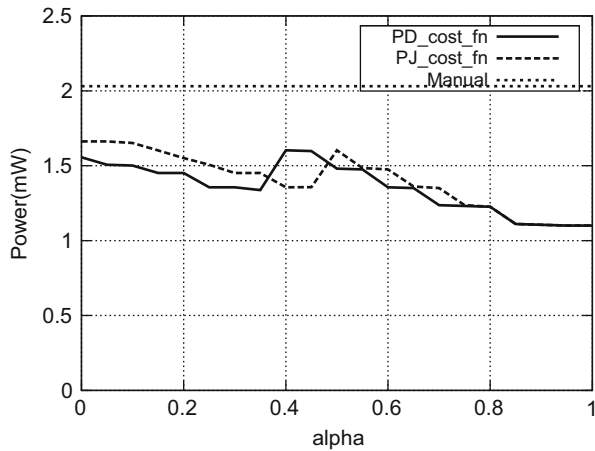
Our DP based algorithm for automated synthesis of the buffered H-tree is described next. We begin by first characterizing (using HSPICE (Inc Meta-Software)) each possible segment configuration, which consists of the different wire-codes, buffer sizes, segment lengths and segment types described earlier in this section. There are a total of 2745 unique segment configurations available to the DP algorithm. For each such segment configuration, we drive it with varying input slews (ranging from 20 ps to 300 ps, with 1 ps increments). We record the jitter, output slew, delay and the power of the segment configuration as a function of its input slew. This characterization data is now used by the DP algorithm. We implement two cost functions for the DP algorithm. Suppose that the cost is being computed for a segment configuration  $i$  which builds upon a partial solution with total jitter  $J$ , total delay  $D$  and total power  $P$ . The segment configuration  $i$  has jitter  $J_i$ , delay  $D_i$  and power  $P_i$ .

- A weighted sum of power and jitter. This cost function is expressed as 
$$C_{PJ} = \alpha \frac{(P+P_i)}{P^*} + (1 - \alpha) \frac{(J+J_i)}{J^*}$$
 Note that  $P + P_i$  and  $J + J_i$  are the total power and total jitter after segment  $i$  is appended to the partial solution. Also note that  $P^*$  and  $J^*$  are the average power and jitter values respectively, for all candidate segment configurations that may be used for segment  $i$ . Since the scale of power and jitter are different, these values are introduced in order to equalize the contribution of power and jitter to the cost function.
- A weighted sum of power and delay (or slew). This cost function is expressed as 
$$C_{PD} = \alpha \frac{(P+P_i)}{P^*} + (1 - \alpha) \frac{(D+D_i)}{D^*}$$
 Note that  $P + P_i$  and  $D + D_i$  are the total power and total delay after segment  $i$  is appended to the partial solution. Also note that  $P^*$  and  $D^*$  are the average power and delay values respectively, for all candidate segment configurations. Since the scale of power and delay are different, these values are introduced in order to equalize the contribution of power and delay to the cost function.

Suppose we have a optimal solution to the buffered H-tree construction problem  $S$ . Clearly, this solution consists of making a choice of the last segment  $s$  (closest to the sink) to be selected. Hence the solution to the buffered H-tree construction problem from the source of the H-tree to the input of segment  $s$  must also be optimal (otherwise we contradict the optimality of  $S$ ). Since the cost functions of jitter, delay and power are additive, an optimal solution to the buffered H-tree construction at any location  $n$  can be constructed from the optimal solution of the buffered H-tree construction problem for every location  $m$  which is downstream from  $n$  (i.e.  $m$  is closer to the source of the H-tree than  $n$ ). Hence the criterion for optimality of DP is satisfied and as a result, DP can yield an optimal solution (for the cost functions of delay, jitter or power) to the buffered H-tree construction problem.

The DP based algorithm selects the lowest cost buffered H-tree implementation that satisfies constraints such as output slew limits and cycle-to-cycle wake-up jitter. The resulting H-trees are simulated in HSPICE, and the end-to-end delay, power and end-to-end jitter estimated by our DP engine are compared with the corresponding values obtained via HSPICE.

**Fig. 2.30** Power (512× Maximum Buffer size)



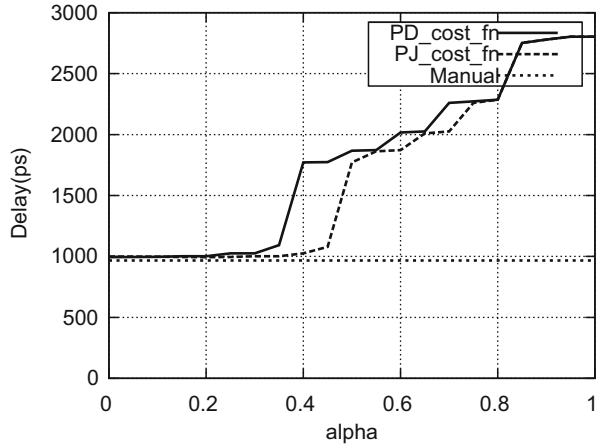
### 2.3.4 Experimental Results

We implemented our DP algorithm in the C programming language. We assumed that the H-tree is implemented in a 45 nm (PTM 2013) technology, with  $VDD = 1.1$  V. All the segment configurations were pre-characterized using HSPICE (Inc Meta-Software). We used the *accurate* option in HSPICE. We set the parameter *delmax* set to 1 ps, for maximum accuracy, which ensures that the maximum timestep that HSPICE takes is limited to 1 ps. We observed that the slew rate at the end of each segment was substantially constant for a given segment configuration, independent of the slew rate at the input to the segment.

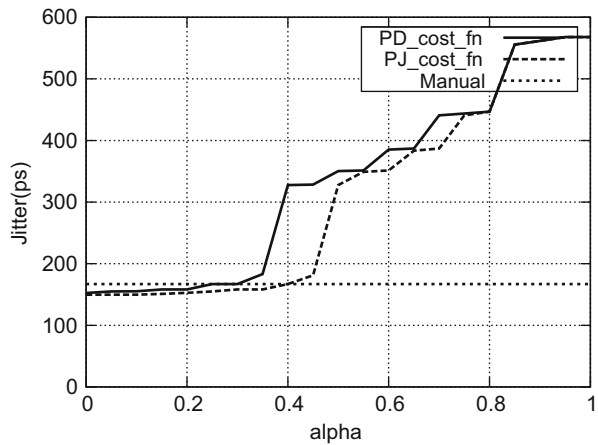
In order to compare our results with a manually generated buffered H-tree, we consulted an industrial clock tree designer and constructed a minimum delay H-tree based on their input. This H-tree utilized the wire-code with the lowest RC delays (WC-2), with the first 6 segments of length 2.5 mm, and the last 2 segments of length 1.25 mm. 512× buffers used were used exclusively to minimize the delay. The DP algorithms' results were compared with the manually generated H-tree, and the comparison is presented next.

We split the experiments into three sets. In the first set, the maximum buffer size allowed was limited to 512×, while in the second and third sets, up to 256× and 128× buffers were allowed respectively. Figures 2.30, 2.31 and 2.32 corresponds to the first set of experiments (with up to 512× buffers). Figures 2.33, 2.34 and 2.35 corresponds to the second set of experiments (with up to 256× buffers). Finally, Figs. 2.36, 2.37 and 2.38 corresponds to the third set of experiments (with up to 128× buffers). We report the power results, delay results, and the jitter results. For each plot, the x-axis corresponds to  $\alpha$ , while the y-axis corresponds to the quantity expressed by the plot. Also, for each plot, the DP results are presented for both cost functions described in Sect. 2.3.3. The results of the manual buffered H-tree are reported in each plot as well.

**Fig. 2.31** Delay (512×  
Maximum Buffer size)



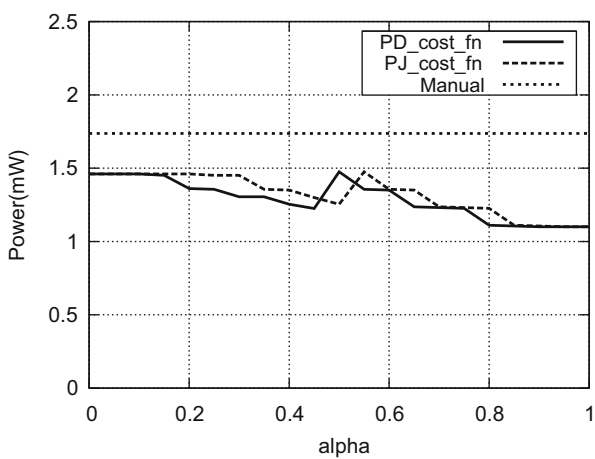
**Fig. 2.32** Jitter (512×  
Maximum Buffer size)



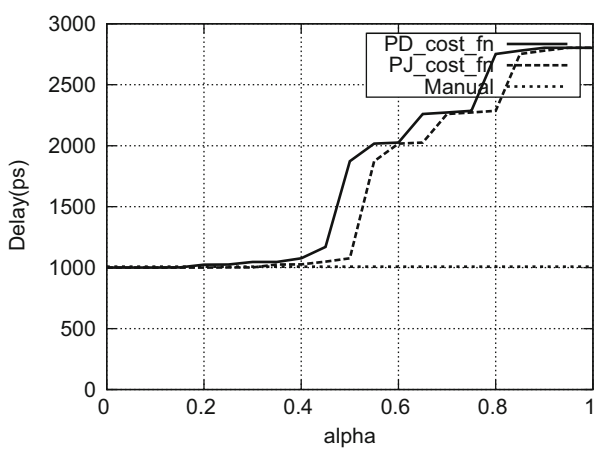
From these plots, we note that for all 3 experiments, the lowest jitter of the buffered H-tree returned by the DP engine is strictly lower than that of the manually synthesized design, by as much as 28 %. This indicates that our DP based buffered H-tree synthesis technique is of value. Further, since our approach is able to select segment configurations freely, it is able to produce strictly better values of power compared to the manual approach, for *all* values of  $\alpha$ . The  $C_{PJ}$  cost function yields lower jitter values compared to the  $C_{PD}$  cost function, especially for  $\alpha > 0.5$ . For values of  $\alpha < 0.5$ ,  $C_{PJ}$  achieves better jitter results than  $C_{PD}$ , but the improvement is lower. However, for power-constrained designs, it may be desirable to use  $\alpha > 0.5$ . The delays of the DP based approaches are higher than the manual design for  $\alpha > 0.5$ , but this is not of importance for PLL based designs, as we have mentioned earlier.

To validate the accuracy of our DP engine, we simulated the buffered H-tree returned by our DP engine (for the cost functions of minimum jitter, minimum

**Fig. 2.33** Power (256× Maximum Buffer size)



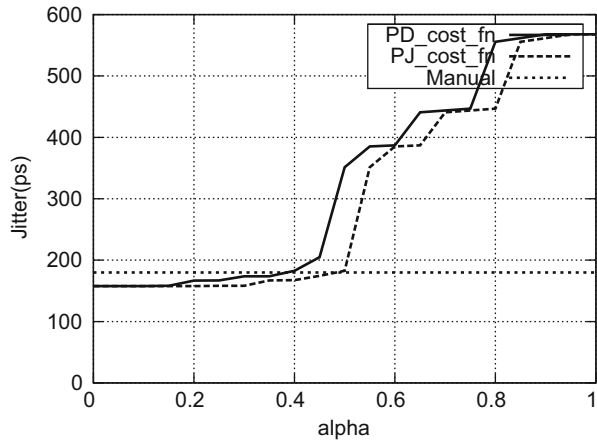
**Fig. 2.34** Delay (256× Maximum Buffer size)



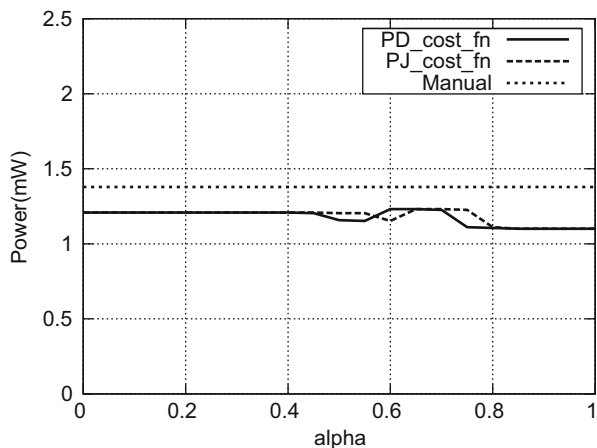
power and minimum delay) in HSPICE. The results were compared for the DP engine running with a maximum allowed buffer size of 128×, 256×, and 512× respectively. Table 2.4 reports the results of this comparison. The delay, jitter and power estimated by our DP engine are reported in Columns 3, 4 and 5 respectively. Columns 6, 7 and 8 report the percentage error in the DP estimate of delay, jitter and power respectively, compared to the HSPICE value. Note that the maximum error in any of these estimates is 4.6 %. The DP’s estimate of these three quantities was always lower than the value returned by HSPICE.

The segments selected by the DP engine (for a maximum allowable buffer size of 512×) are reported in Tables 2.5 and 2.6 (for the cost function of minimum jitter and minimum delay respectively). Segments with lower indices (Column 1) are closer to the H-tree source. For each segment, we report the segment index, wire-code (Column 2), segment length in mm (Column 3), segment style (one of Straight (S),

**Fig. 2.35** Jitter ( $256\times$   
Maximum Buffer size)



**Fig. 2.36** Power ( $128\times$   
Maximum Buffer size)

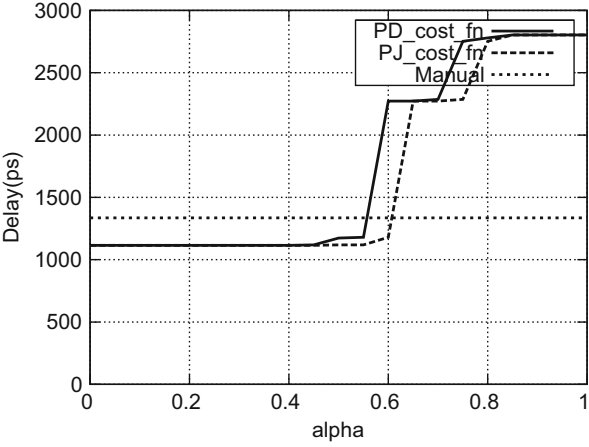


$T$  or  $I$ ) (Column 4) and buffer size (Column 5). Note that the segment lengths for Tables 2.5 and 2.6 are quite different, indicating that the minimum delay buffered H-tree is not the same as minimum jitter buffered H-tree. Also, only 2 of the wire-codes end up getting used. All segments use a  $512\times$  buffer since power is not constrained.

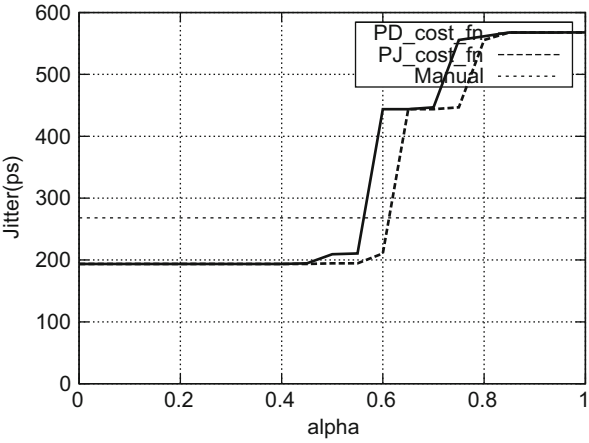
### 2.3.5 Conclusion

Buffered clock distribution networks have become increasingly popular due to increasing on-chip wiring delays. Since clock buffers are liable to add jitter in the clock signal on account of cycle-to-cycle  $VDD$  variations in the die, we argue that the design goal of minimizing end-to-end jitter is more relevant for buffered H-tree synthesis than minimizing end-to-end delay (which is irrelevant for PLL based designs). In this

**Fig. 2.37** Delay (128×  
Maximum Buffer size)



**Fig. 2.38** Jitter (128×  
Maximum Buffer size)



work, we therefore present a dynamic programming based approach to synthesize a minimum cost buffered H-tree clock distribution network. Our cost functions are a weighted sum of power and jitter, Juniper Networks (2010) or a weighted sum of power and end-to-end delay of the distribution network. After pre-characterizing the delay, jitter and power of buffered segments (2745 in all) of different lengths, topologies, buffer sizes and wire-codes, a dynamic programming (DP) engine automatically generates the optimal H-tree that minimizes the appropriate cost function. Compared to a manually constructed buffered H-tree network, our approaches are able to reduce both jitter (by as much as 28 %) and power (by as much as 46 %). When optimizing for minimum jitter, the DP engine generates a H-tree with lower jitter than when optimizing for minimum delay, thereby validating our approach, and proving its utility.

**Table 2.4** Comparison of DP results with HSPICE

		DP Estimate			Difference versus HSPICE (%)		
	Cost Func.	Delay (ps)	Jitter (ps)	Power (mW)	Delay (ps)	Jitter (ps)	Power (mW)
Up to 128×	Min. Delay	1114.7	193.5	1.21	0.59	1.88	4.58
	Min. Power	1334.9	247.6	1.08	0.57	1.94	2.44
	Min. Jitter	1114.7	193.5	1.21	0.59	1.88	4.58
Up to 256×	Min. Delay	1000.1	157.7	1.46	0.80	2.23	4.58
	Min. Power	1334.9	247.6	1.08	0.57	1.94	2.44
	Min. Jitter	1000.1	157.7	1.46	0.80	2.23	4.58
Up to 512×	Min. Delay	994.2	152.0	1.55	0.89	2.69	2.52
	Min. Power	1334.9	247.6	1.08	0.57	1.94	2.44
	Min. Jitter	999.3	149.0	1.66	1.11	2.17	2.92

**Table 2.5** Segments selected by Our DP Algorithm (Min. Jitter, 512× max)

Seg. #	Wire-code	Length (mm)	Seg. type	Buf. size
1	WC1	1	I	512×
2	WC2	2	S	512×
3	WC2	2	S	512×
4	WC1	1	I	512×
5	WC2	2	S	512×
6	WC2	2	S	512×
7	WC1	1	I	512×
8	WC2	1.5	S	512×
9	WC1	1	I	512×
10	WC2	1.5	S	512×
11	WC2	1.25	I	512×
12	WC2	1.25	I	512×

**Table 2.6** Segments selected by our DP Algorithm (Minimum Delay, 512× Maximum)

Seg. #	Wire-code	Length (mm)	Seg. type	Buf. size
1	WC1	1	I	512×
2	WC2	1.75	S	512×
3	WC2	2.25	S	512×
4	WC1	1	I	512×
5	WC2	1.75	S	512×
6	WC2	2.25	S	512×
7	WC1	1	I	512×
8	WC2	1.5	S	512×
9	WC1	1	I	512×
10	WC2	1.5	S	512×
11	WC2	1.25	I	512×
12	WC2	1.25	I	512×

2.4 Tiled SWO-based Clock Distribution

In the above section, we present a dynamic programming based approach to synthesize a minimum cost buffered H-tree clock distribution network. However, in Sect. 2.2, we observe that buffered H-tree suffers from a high jitter. Moreover,



buffered H-tree has a high power consumption at higher frequencies. In order to address the above issues, we explore a SWO-based high-speed clock distribution in the following section. We validate that a SWO approach can be used to practically implement a high-frequency, low-power, low jitter clocking approach with high and uniform area coverage over an IC.

### 2.4.1 Introduction

As discussed in Sect. 2.1, there has been much interest in ring-based resonant oscillators as a means to generate the clock signal in digital ICs. The parasitic inductance and capacitance of the rings are fixed once the ring perimeter, wire dimensions, layer and spacing are determined. The oscillation frequency of a resonant oscillator is determined by the parasitic inductance and capacitance values, provided the inverter pair(s) can switch at this frequency. The equivalent circuit for such a resonant oscillator is shown in Fig. 2.6. In this figure, the parasitic inductance of the ring is referred to as  $L_w$ . The parasitic capacitance of the ring is called  $C_w$ . The capacitance due to the cross-coupled inverter pair (i.e. twice the sum of the diffusion and gate capacitances of any inverter in the pair) is  $C$ . Since  $C$  and  $C_w$  are in parallel, we obtain the equivalent circuit shown. The oscillation frequency of the equivalent circuit is given by Eq. 2.1.

Although the resonant oscillator structure has tremendous potential as a means to generate a high-frequency on-chip clock signal with low power consumption, it has one fundamental drawback. To enable high-frequency oscillations, the parasitic inductance and capacitance of the ring need to be held at low values, which means that the ring perimeter necessarily needs to be small. Typical values of the ring perimeter are  $\sim 2$  mm. Since ICs can be as large as 20–30 mm on a side, the resonant clocking idea cannot practically be used to generate a high-frequency *chip – wide* clock signal. This key focus of this section is to address this issue. Since a SWO Cordero and Khatri (2008) generates a clock signal which has identical phase at each point along the ring (unlike a TWO), we focus our attention on SWOs.

An SWO with a large area coverage on the IC die can be generated in one of two ways.

- **Option A: Implement a  $k \cdot \lambda/2$  ring:** Consider an SWO with perimeter  $p$ . If we traverse such a ring once, the total phase change is  $\lambda/2$ . To ensure a larger area coverage, we can increase the size of the ring to  $k \cdot p$ , where  $k$  is odd, while forcing the total phase change over a single traversal of the ring to be  $k \cdot \lambda/2$ . This approach does not compromise oscillation frequency, and grows the area coverage of the clock signal by a factor of  $\sqrt{k}$ . This approach was used in Sect. 2.2.
- **Option B: Tile several SWO on a 2D plane:** Another approach is to arrange several identical SWO rings in a 2D tiled structure. Each SWO ring oscillates at the same frequency. We additionally force each adjacent ring to oscillate with an identical phase.

Even though we can set  $k$  to a very high value in principle, the first option may be impractical since it realizes a single ring with a very large perimeter, with no clock coverage in the regions in the center of the ring. We overcome this limitation in Sect. 2.2 by snaking the wires of the ring to ensure a uniform clock coverage.

The second approach solves the uniform clock coverage issue, but since each individual SWO has a small perimeter, the second approach alone would require a large number of SWO rings to be implemented. For example, with a chip of size 1 cm on a side and a ring perimeter of 2 mm, 400 SWO rings would be required, making the approach impractical.

In this section, we propose to use a combination of the above two approaches to achieve a large area coverage for the clock signal, in a uniform manner. We present our approach by means of an example in which  $k = 3$ , and a  $3 \times 3$  tiling structure is used. We show that  $k = 3$  is a good choice since it results in an elegant 2D embedding of the SWO tiles. We have experimented with  $k = 5, 7$  and  $9$  as well, and have validated correct operation of such SWOs. Also, our tiling structure can be easily generalized to an arbitrarily large  $n \times n$  arrangement of SWO tiles. With  $k = 3$ , and a chip of size 1 cm on a side, the number of required SWO rings for complete chip coverage is reduced by a factor of 8, compared to the case where  $k = 1$ .

The remainder of this section is organized as follows: Previous work is described in Sect. 2.4.2, while Sect. 2.4.3 provides the details of our tiled high-speed, low-power and high area coverage clock distribution strategy using SWOs. In Sect. 2.4.4 we present results from experiments which we conducted to validate our approach. We conclude in Sect. 2.4.5.

## 2.4.2 Previous Work

In Sect. 2.1, we introduced resonant ring-based oscillators. A traveling wave resonant oscillator (TWO) circuit (referred to by the authors as a *rotary clock*) was described and implemented (Wood et al. 2001; 2006). The key idea in this approach is to utilize a sufficiently long wiring ring, such that its capacitive and inductive parasitics result in a high frequency oscillatory network. The main problem with a TWO is the phase change incurred around the ring. In response to this, a standing wave resonant oscillator circuit was proposed (Cordero and Khatri 2008). The clock signal at any point in the ring is sinusoidal, but has the *same phase* at all points along the ring. To recover a full-rail clock anywhere along the ring, differential amplifiers need to be connected to the ring signals at these locations.

In (Mahony et al. 2003; Karkala et al. 2009), a high-frequency standing wave oscillator was used to implement a clock Phase Locked Loop (PLL). The work of (Mahony et al. 2003) is based on the use of multiple coupled oscillators (each comprised of an NMOS cross-coupled pair to sustain the oscillation, and a PMOS diode connected load for setting the common mode voltage). The approach of Karkala et al. (2009) implements a resonant SWO based PLL, with an inductance control based coarse frequency adjustment mechanism. Fine frequency adjustment is achieved by

controlling the body bias of the PMOS transistor of the inverter pair. Unlike the approach proposed in this work, these approaches did not address the key problem of the IC area coverage of resonant SWOs or TWOs. This work has the ability to cover large die areas with a high-frequency, low power clock signal by using interlinked tiled SWOs, each of which have a total phase change of  $3\lambda/2$  across the ring.

In Mahony et al. (2003), the authors present a tiled SWO based resonant grid for high frequency clock distribution. Each SWO is implemented as  $\lambda/2$  ring, using a short circuit at the far end (instead of a mobius connection as in our case). Multiple SWOs are coupled by injection locking. The key differences between (Mahony et al. 2003) and our approach are i) we utilize  $3\lambda/2$  rings (which results in fewer SWOs being required) and ii) we utilize a mobius termination in each SWO ring, while (Mahony et al. 2003) utilizes a short circuit termination for each SWO ring. It was shown (Cordero and Khatri 2008) that short circuit termination results in a lower oscillation frequency as well as higher power in comparison to a mobius termination based SWO.

### 2.4.3 Our Approach

Resonant oscillators (SWOs as well as TWOs) are a promising technique to generate a high-frequency on-chip clock signal with low power. However, they possess a key weakness when used in typical ICs, where the goal is to uniformly distribute a chip-wide, high-frequency, low power clock signal. To achieve a high frequency of operation, the typical values of inductance and capacitance required for the resonant oscillators are such that the total perimeter of the resonant ring is small (typically  $\sim 2$  mm). Since many complex ICs can be as large as 20-30 mm on a side, the ratio of the chip area to the area covered by a typical resonant ring is as high as  $\sim 3600$ . Hence it would be impossible to distribute a chip-wide, high-frequency resonant clock signal with a single SWO or TWO ring. Our goal is to present approaches to achieve complete and uniform area coverage of the resonant clock signal across the IC die. By uniform area coverage, we mean that at any position on the IC die, a resonant clock signal is no further away than the perimeter of an individual resonant ring (i.e.  $\sim 2$  mm). Since a SWO (Cordero and Khatri 2008; Karkala et al. 2009) generates a clock signal which has identical phase at each point along the ring (unlike a TWO), we focus our attention on SWOs.

From the equivalent circuit for our resonant oscillator (Fig. 2.6) and the equation for the oscillation frequency of the resonant oscillator (Eq. 2.1), we observe that increasing the perimeter of the ring is not an acceptable option to achieve high clock coverage on the IC die. This is because increasing the perimeter of the ring increases both  $C_w$  and  $L_w$  linearly, resulting in an unacceptable drop in clock frequency. As a result, we explore two alternative options:

**Option A:** For an SWO with perimeter  $p$ , if we traverse the ring once, the total phase change is  $\lambda/2$ . To ensure a larger area coverage, we can increase the perimeter of the ring to  $k \cdot p$ , where  $k$  is odd, thereby making the total phase change over a

single traversal of the ring to be  $k \cdot \lambda/2$ . In such a design, we require  $p$  equally spaced inverter pairs, and an odd number (typically one) of mobius connections. The circuit configuration for a  $3 \cdot \lambda/2$  ring is shown in Fig. 2.7. Note that in order to ensure that the resonant structure bootstraps in a standing wave configuration, the signals at the inverter pair are initialized using a global bootstrap signal (labeled *BS* in Fig. 2.7). We have validated that the  $k \cdot \lambda/2$  SWO ring oscillates correctly and reliably, and at the same frequency as the corresponding  $\lambda/2$  ring, for  $k = 3, 5, 7$  and 9.

Although the  $k \cdot \lambda/2$  approach does not compromise oscillation frequency, and also increases the area coverage of the clock signal by a factor of  $\sqrt{k}$ , it still possesses a significant drawback. For large ICs, the value of  $k$  needs to be significantly large. For example, for a chip of size 30 mm on a side, a  $k$  value of about 61 is required. With such a configuration, a key problem is the uniformity of the coverage of the clock across the die. The center of the ring in such a case is about 15 mm from the nearest resonant clock location, making the approach impractical. Implementing the resonant SWO ring in a snaked manner (as designed in Sect. 2.2) is one solution to this problem.

**Option B:** Another approach is to arrange several identical  $\lambda/2$  SWO rings in a 2D tiled structure. Each SWO ring oscillates at the same frequency. We additionally force adjacent rings to oscillate with an identical phase by introducing an appropriate number of shorts across these rings. Suppose our chip size is 10 mm on a side. In this case, assuming  $\lambda/2 = 2$  mm, we would require 400 SWO rings. The advantage of this approach is that it enables us to achieve a uniform and complete area coverage of the clock signal on the IC die, but the drawback is that we need a large number of SWOs.

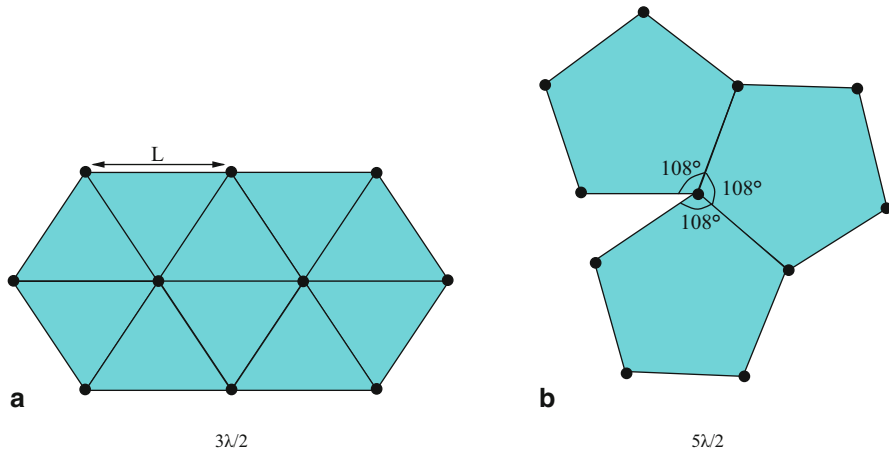
**Option C:** The third option is a hybrid of the Options A and B. In this case, we arrange several identical  $k\lambda/2$  SWO rings in a 2D tiled structure. This approach therefore retains the best features of both Options A and B. This work utilizes Option C (with  $k = 3$ ) to implement a complete and uniform chip-wide resonant clock distribution network.

Next, we discuss the details of our approach.

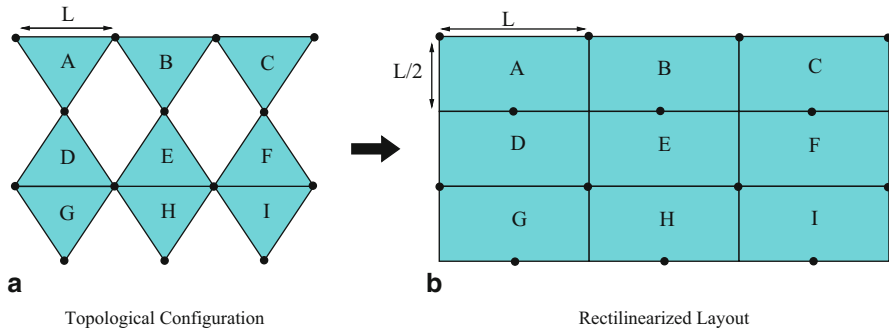
### 2.4.3.1 Tiled SWO Topology

For a tiled  $k\lambda/2$  SWO, we first need to choose the value of  $k$ . The problem becomes that of embedding a regular  $k$ -sided polygon on a plane. This is illustrated via Fig. 2.39.

The internal angle of a regular  $k$ -sided polygon is given by  $Z = \frac{(n-2) \cdot 180^\circ}{n}$ . In order that the  $k$ -sided polygon can be embedded on a plane, we require that  $nZ = 360^\circ$ , where  $n$  is an integer. Given that  $k$  is odd, the only value of  $k$  that satisfies the above condition is 3. Hence we choose  $k = 3$ . Figure 2.39 illustrates how a uniform triangle can be embedded on a plane, while a uniform pentagon cannot. Note that each dot in Fig. 2.39a represents six inverter pairs (one for each SWO ring).



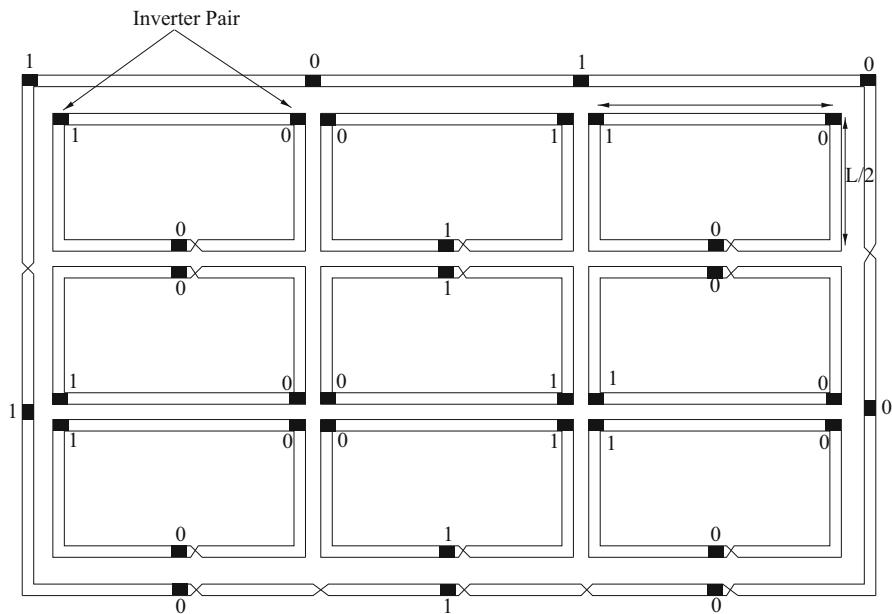
**Fig. 2.39** Embedding a triangle and a pentagon on a 2D plane



**Fig. 2.40** Rectilinear realization of a triangle on a 2D plane

Note that the embedding of the equilateral triangle on a 2D plane shown in Fig. 2.39 cannot be directly implemented in a VLSI IC, since wires on an IC are constrained to be rectilinear. In order to perform the embedding of a  $3\lambda/2$  SWO ring on a 2D surface (using rectilinear wires), we first remove every alternate SWO ring. Now the resulting structure (shown in Fig. 2.40a) has half as many SWO rings. Each dot in this figure represents 2 inverter pairs in the even numbered rows, and 4 inverter pairs in the odd numbered rows. We rectilinearize the segments of Fig. 2.40a, and the result is shown in Fig. 2.40b.

In order to achieve the rectilinearized embedding of  $3\lambda/2$  SWO rings, we transform each non-rectilinear wire of the embedding of Fig. 2.40a and convert it into a single wire with a horizontal and a vertical segment. Thus each  $3\lambda/2$  ring is transformed into a rectangle with length  $L$  and height  $L/2$ , where  $L$  is the perimeter of the corresponding  $\lambda/2$  ring. Each edge in Fig. 2.40b represents 4 wire segments, with one pair of wire segments being utilized by each of the two separate  $3\lambda/2$  rings which share the edge. Each dot of Fig. 2.40b represents 2 inverter pairs, with each inverter pair being utilized by the two separate  $3\lambda/2$  rings which share the dot.



**Fig. 2.41** Layout organization of a 3x3 SWO tile

A more detailed view of the tiled  $3\lambda/2$  SWO rings (for a  $3 \times 3$  tiled array) is shown in Fig. 2.41. Each ring consists of 2 (inner) wires, with two outer wires corresponding to rings that are above, below, or on either side of the said ring. We need to make all rings oscillate with the same frequency and phase. In addition, we need to ensure that all 4 wires of any segment, at any location, have the same voltage at any time instant. In order to guarantee these conditions, we utilize bootstrap devices, to force initial conditions at various locations along the tiled SWO structure. Although the bootstrapping devices are not shown in Fig. 2.41, the values that are asserted at various locations in the ring by these devices are shown (by means of the '0' and '1' labels). In order to guaranteed that all rings oscillate with the same frequency and phase, it is crucial to ensure that the electrical environment around each location of any ring is identical to the electrical environment around the same location of all other rings. In order to do this, we insert an outermost peripheral ring as well, whose length is  $9L$ . This ring also oscillates, ensuring that the electrical environment of each tiles is identical. Note that the mobius connections of each of the tiles are illustrated in Fig. 2.41. The outer ring has 4 extra mobius flips (in addition to those required to sustain oscillations) shown along its lower edge. These flips are introduced in order to ensure that every location of the outer wire oscillates with the same frequency, phase and amplitude as the wire in the SWO tile adjoining it. We experimented with several ways of connecting the outer ring (such as grounding it at regular intervals and leaving it floating), and found that in order to ensure low jitter and uniform oscillation frequency, it was essential to connect the outer ring in the configuration shown in Fig. 2.41.

### 2.4.4 Experiments

We implemented the tiled SWO described in this work, using a 90 nm BSIM3 (PTM 2013) process technology. The power supply voltage was 1.2 V, and all simulations were conducted in HSPICE (Inc Meta-Software). We simulated a  $3 \times 3$  tiling structure (as described in Fig. 2.41). The ring consisted of 7 wires in all, each of which had a width of  $20 \mu\text{m}$  and a inter-wire spacing of  $20 \mu\text{m}$  as well. The outermost and innermost wires as well as the middle wire are connected to ground, with the remaining wires are utilized to carry the 4 oscillating signals. The RLC parasitics of the 7-wire bundle were extracted using (Raphael Interconnect Analysis Tool: User's Guide), and adjusted for skin-effect in our simulations. The nominal oscillation frequency of each of the 9 SWO rings was 7.267 GHz (yielding a nominal clock period  $T_{nom} = 137.6$  ps. Each SWO ring consists of 72 smaller segments in our HSPICE simulation deck.

The two wires of any SWO ring sustain a sinusoidal oscillation. To recover a rail-to-rail clock from any point on the ring, a clock recovery circuit (shown in Fig. 2.3) is required. This circuit is essentially a differential amplifier with a buffered output. We implemented 42 regenerator circuits per SWO ring. An overlay plot of all  $42 \times 9$  recovered signals is shown in Fig. 2.42. From this figure, we observe that the falling skew is 4.56 ps, while the rising skew is 1.45 ps (for a clock of period of 137.6 ps).

The power consumption of our oscillator is 55.07 mW per SWO ring (without any regenerators) and 68.56 mW per SWO ring (with 42 regenerators per SWO ring). This would indicate that for a chip with size 10 mm per side, the total power consumption in the clock distribution network would be  $\sim 2.75$  (3.43) Watts without (with) regenerators.

To measure the quality of the tiled SWO based clock distribution network, we report several quantities obtained after simulating the structure for 140 clock cycles. These quantities serve as figures of merit of the design, and are listed below:

- We computed, at each of the 27 inverter pair sites (3 for each of the 9 SWO rings), the clock period for each clock cycle. Let  $T_{max}$  and  $T_{min}$  be the maximum and minimum clock periods, and  $\Delta_T = T_{max} - T_{min}$ . The worst case value of  $\frac{\Delta_T}{T_{nom}}$  over all the 27 inverter pair locations was 1.56 % (measured at the ring) and 2.61 % (measured after the regenerators).
- Recall that the location between two inverter pairs is a virtual ground location. Therefore the amplitude of the sinusoidal signal on either side of the virtual ground location is small, making it hard to reliably regenerate a square wave clock from the ring locations on either side of the virtual ground. In our experiments, we did not connect regenerators to ring locations which were within .345 mm on either side of a virtual ground node (which yielded 42 regenerators per ring, for a total of  $42 \times 9 = 378$  regenerators). We found that the worst case value of  $\frac{\Delta_T}{T_{nom}}$  over all points (where the clock can be extracted) over all rings was 1.89 % (measured at the ring) and 3.13 % (measured after the regenerators).

Figure 2.43 displays an overlay plot of 3 virtual ground nodes (for rings A, G and H as labeled in Fig. 2.40b). The virtual ground waveforms have a peak-to-peak voltage

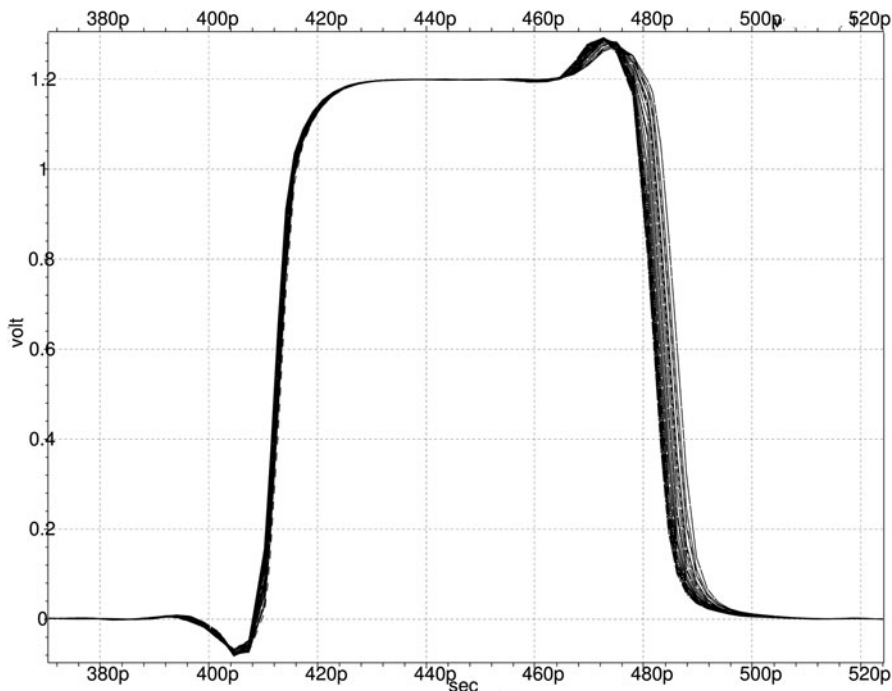


Fig. 2.42 Overlay of recovered waveforms from all 378 regenerators

of about 200 mV over all 27 virtual ground locations. Finally, Fig. 2.44 displays the overlay plot of the SWO ring waveforms (for ring A). The waveforms correspond to all 24 internal ring nodes encountered between two adjacent inverter pairs of ring A.

We also computed the Q factor for any of the 9 rings of our tiled SWO oscillator. To compute the Q factor, we first removed the inverter pairs of the ring, and replace them by the equivalent average capacitance of the inverter pair terminals. Now an differential AC current with differential amplitude of 1 A is applied across these terminals. The resulting voltage across the terminals is measured as a function of the frequency of the differential current. The voltage has a peak at the oscillation frequency of the ring. The Q factor is computed by finding the ratio of the resonant frequency to the 3dB bandwidth of the voltage waveform. We determined the Q factor of our tiled SWO oscillator to be about 19.

### 2.4.5 Conclusion

Resonant oscillators can sustain extremely high oscillation frequencies with very low power consumption. However, a single resonant oscillator covers a very small fraction of the area of a typical IC. In this work, we present an approach to completely and uniformly cover an IC using an SWO. This is achieved by combining two



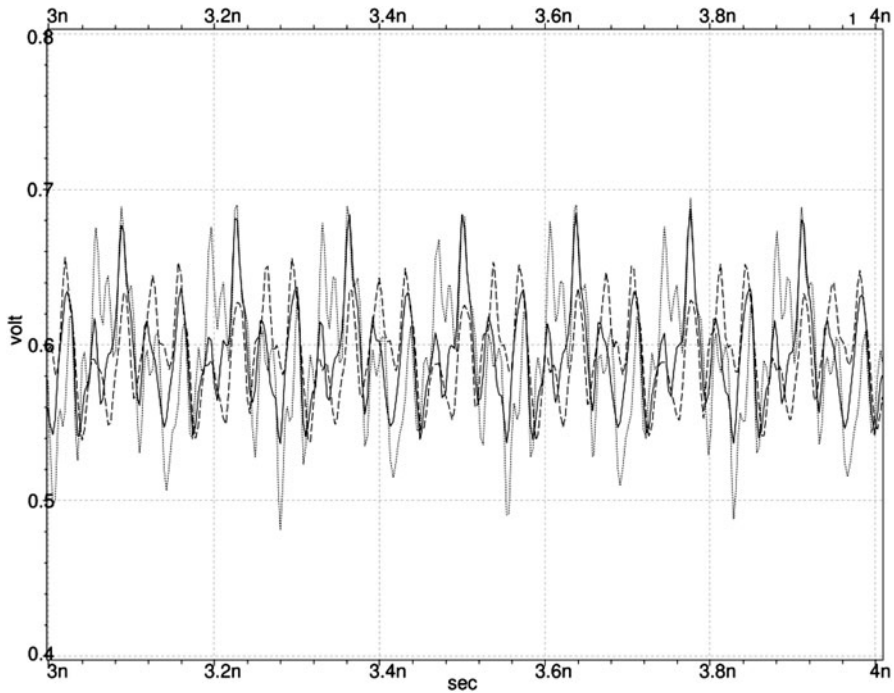
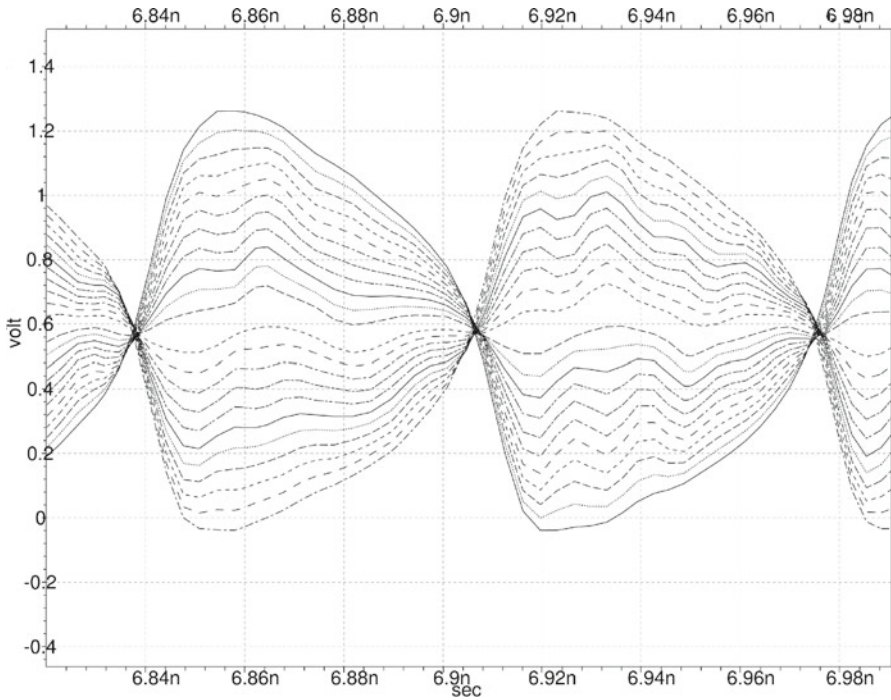


Fig. 2.43 3 Overlaid virtual ground waveforms

techniques. The first technique increases the area coverage of an individual SWO by ensuring that it sustains 3 standing waves along the ring. The second approach further increases the area coverage by tiling multiple such SWOs side by side, and connecting them such that they oscillate with the same high frequency and phase. We carefully ensure that the electrical environment around each SWO ring is identical. Skin effect adjusted 3D RLC parasitics are utilized for our experiments. For a 90 nm process, our tiled SWO based resonant clock distribution approach achieves an oscillation frequency of about 7.267 GHz, with a low power consumption of about 68.5 mW per SWO ring, and a jitter of 3.1 % of the nominal clock period.

## 2.5 Chapter Summary

In this chapter, we address the issue of distributing a high-speed, low power, low jitter clock with the aim of serving the NoC and the PEs in a CMP. In Sect. 2.1, we introduce resonant ring-based oscillators, which have recently emerged as a promising technique for high-speed, low power clock generation. In Sect. 2.2, we use a resonant SWO based phase-locked clock generation and distribution scheme with superior jitter, skew and power characteristics. Phase locking is accomplished



**Fig. 2.44** Overlaid waveforms of ring signals between 2 Adjacent inverter Pairs

by a fully digital control loop consisting of a coarse frequency control and a fine frequency control. The SWO frequency is modulated in a coarse manner by changing the ring inductance, by modifying the number of oscillating wires. Fine frequency control is achieved by modifying ring capacitance via a binary-weighted capacitor bank. Clock distribution is performed by routing the resonant ring over the die in a “comb” manner. Our approach is compared to an H-tree with an overlaid mesh. Given the relatively high Q-factor of the SWO, this clocking approach exhibits a very tight cycle-to-cycle jitter (about  $4.6\times$  better) and very low skew (about  $5.5\times$  better). The power consumption of our scheme is upwards of  $2\times$  improved as well, given the resonant nature of the approach.

In recent fabrication technologies, increasing on-chip wiring delays have contributed to the popularity of buffered clock distribution network. In Sect. 2.3, we present a dynamic programming based approach to synthesize a minimum cost buffered H-tree clock distribution network. Our two cost functions are a weighted sum of power and jitter, and a weighted sum of power and end-to-end delay of the distribution network respectively. After pre-characterizing the delay, jitter and power of buffered segments (2745 in all) of different lengths, topologies, buffer sizes and wire-codes, we invoke a dynamic programming (DP) engine to automatically generate the optimal H-tree that minimizes the appropriate cost function. Compared to a manually constructed buffered H-tree network, our approaches are able to reduce

both jitter (by as much as 28 %), and power (by as much as 46 %). When optimizing for minimum jitter, the DP engine generates a H-tree with lower jitter than when optimizing for minimum delay, thereby validating our approach, and proving its utility.

In general, SWO based clock distribution networks can suffer from a non-uniform clock coverage at different die locations. In order to address this issue, we explore a tiled SWO-based high-speed clock distribution in Sect. 2.4. We validate that a SWO approach can be used to practically implement a high-frequency, low-power, low jitter clocking approach with high and uniform area coverage over an IC. This is achieved by combining two techniques. The first technique increases the area coverage of an individual SWO by ensuring that it sustains 3 standing waves along the ring. The second approach further increases the area coverage by tiling multiple SWOs side by side, and connecting them such that they oscillate with the same high frequency and phase. We carefully ensure that the electrical environment around each SWO ring is identical. Skin effect adjusted 3D RLC parasitics are utilized for our experiments. Our tiled SWO based resonant clock distribution approach achieves an oscillation frequency of about 7.267 GHz, with a low power consumption of about 68.5 mW per SWO ring, and a jitter of 3.1 % of the nominal clock period.

## References

- F. Anceau, "A synchronous approach for clocking VLSI systems," *Solid-State Circuits, IEEE Journal of*, vol. 17, no. 1, pp. 51-56, Feb 1982.
- Y Chen and D. F. Wong, "An algorithm for zero-skew clock tree routing with buffer insertion," *Design and Test of Computers*, pp. 230-236, 1996.
- S.C. Chan, P.J. Restle, K.L. Shepard, N.K. James, and R.L. Franch, "A 4.6 GHz resonant global clock distribution network," *Solid-State Circuits Conference, 2004. Digest of Technical Papers. ISSCC. 2004 IEEE International*, pp. 342-343 Vol.1, Feb. 2004.
- T-H Chao, Y-C Hsu, J-M Ho, and A. B. Kahng, "Zero skew clock routing with minimum wirelength," *IEEE Transactions on Circuits and Systems II*, vol. 39, pp. 779-814, Nov 1992.
- R. Chaturvedi and J. Hu, "Buffered clock tree for high quality IC design," in *Proceedings, International Symposium on Quality Electronic Design*, 2004, pp. 381-386.
- J-Y Chueh, C Ziesler, and M Papaefthymiou, "Experimental evaluation of resonant clock distribution," in *Proceedings, IEEE Computer society Annual Symposium on VLSI*, 2004, Feb 2004, pp. 135-140.
- J Cong et al., "Bounded-skew clock and Steiner routing," *ACM Transactions on Design Automation of Electronic Systems*, vol. 3, pp. 341-388, 1998.
- V Cordero and S Khatri, "Clock distribution scheme using coplanar transmission lines," in *DATE*, 2008, pp. 985-990.
- E.G. Friedman, "Clock distribution networks in synchronous digital integrated circuits," *Proceedings of the IEEE*, vol. 89, no. 5, pp. 665-692, may 2001.
- Juniper Networks San Jose CA N Jayakumar, VLSI Design Engineer, "Private Communication," July 2010.
- Inc Meta-Software, "HSPICE user's manual," Campbell, CA.
- V. Karkala, K.C. Bollapalli, R. Garg, and S.P. Khatri, "A PLL design based on a standing wave resonant oscillator," in *IEEE International Conference on Computer Design (ICCD) 2009*, Oct 2009, pp. 511-516.

- T. H. Lin and W. J. Kaiser, "A 900-MHz 2.5-mA CMOS Frequency Synthesizer with an Automatic SC Tuning Loop," *IEEE Journal of Solid-State Circuits*, vol. 36, no. 3, pp. 424-431, 2001.
- I. Liu, T. Chou, A. Aziz, and D. F. Wong, "Zero-skew clock tree construction by simultaneous routing, wire sizing and buffer insertion," in *Proceedings, Intl Symposium on Physical Design*, 2000, pp. 33-38.
- T. H. Lin and Y. J. Lai, "An Agile VCO Frequency Calibration Technique for a 10-GHz CMOS PLL," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 2, pp. 340-349, 2007.
- F. O'Mahony, "10 GHz Global Clock Distribution using Coupled Standing-Wave Oscillators," Ph.D. dissertation, Stanford University, 2003.
- F. O'Mahony, P. Yue, M. Horowitz, and S. Wong, "Design of a 10GHz clock distribution network using coupled standing-wave oscillators," in *DAC '03: Proceedings of the 40th conference on Design automation*. 2003, pp. 682-687, ACM.
- "MultiGig," <http://www.multigig.com> (Accessed April 22, 2013).
- N. Nedovic, N. Tzartzanis, H. Tamura, F. M. Rotella, et al., M. Wiklund "A 40-44 Gb/s 3 X Oversampling CMOS CDR/1:16 DEMUX," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 12, pp. 2726-2735, 2007.
- "PTM website," <http://www.eas.asu.edu/~ptm> (Accessed April 22, 2013).
- A. Rajaram and D. Pan, "Variation tolerant buffered clock network synthesis with cross links," *Proceedings, Intl Symposium on Physical Design*, pp. 157-164, 2006.
- "Raphael Interconnect Analysis Tool: User's Guide," .
- W. B. Wilson, U. K. Moon, K. Lakshmikummar, and L. Dai, "A CMOS Self-Calibrating Frequency Synthesizer," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 10, pp. 1437-1444, 2000.
- Wood et al. 2001 J, T Edwards, and S Lipa, "Rotary traveling-wave oscillator arrays: a new clock technology," *IEEE Journal of Solid-State Circuits*, vol. 36, no. 11, pp. 1654-1665, Nov 2001.
- J. Wood, T. Edwards, and C. Ziesler, "A 3.5 GHz Rotary-Traveling-Wave-Oscillator Clocked Dynamic Logic Family in 0.25  $\mu$ m CMOS," *Solid-State Circuits Conference, 2006. ISSCC 2006. Digest of Technical Papers. IEEE International*, pp. 1550-1557, Feb. 2006.
- itrs "The International Technology Roadmap for Semiconductors," <http://public.itrs.net/> (Accessed April 22, 2013), 2007.
- J-L Tsai, T-H Chen, and C Chen, "Zero skew clock-tree optimization with buffer insertion/sizing and wire sizing," *IEEE Transactions on CAD*, vol. 23, pp. 565-572, 2004.

Source-Synchronous Networks-On-Chip

Circuit and Architectural Interconnect Modeling

Mandal, A.; Khatri, S.P.; Mahapatra, R.

2014, XIII, 143 p. 95 illus., 10 illus. in color., Hardcover

ISBN: 978-1-4614-9404-1