Title

CMCS: Current-Mode Clock Synthesis

Journal

IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 25(3)

ISSN

1063-8210

Authors

Islam, Riadul
Guthaus, Matthew R

Publication Date

2017-03-01

DOI

10.1109/tvlsi.2016.2605580

CMCS: Current-Mode Clock Synthesis

Riadul Islam, *Student Member, IEEE*, and Matthew R. Guthaus, *Senior Member, IEEE*

***Abstract*—In a high performance VLSI design, the clock network consumes a significant amount of power. While most existing methodologies use voltage-mode (VM) signaling, these clock distributions lose a tremendous amount of dynamic power to charge/discharge the large global clock capacitance. New circuit approaches for current-mode (CM) clocking save significant clock power, but have been limited to only symmetric networks, while most application specific integrated circuits (ASICs) have asymmetric clock distributions. In this paper, we propose the first CM clock synthesis (CMCS) methodology to reduce overall clock network power with low skew. The method can integrate with traditional clock routing followed by transmitter and receiver sizing. We validate the proposed methodology using ISPD 2009 and 2010 industrial benchmarks using an extracted spice model distributed in $1.4 - 275.6mm^2$ area and consists of 81-2249 sinks. This methodology saves $39 - 84\%$ average power with similar skew on the benchmarks using $45nm$ CMOS technology simulation of clock frequencies range from 1-3GHz. In addition, the CMCS methodology takes $2.4 - 9.1\times$ less running time and consumes $20 - 26\%$ less transistor area compared to synthesized, buffered VM clock distributions.**

***Index Terms*—Current-mode, flip-flop, clock distribution network, clock skew, low-power.**

I. Introduction

Clock distribution networks (CDNs) have a tremendous impact on overall dynamic power and performance in VLSI systems. As technology progresses, the complications associated with distributing the CDN are becoming increasingly more challenging.

Many researchers have already proposed different ways to reduce CDN power [1]–[7]. In addition to power, a tremendous amount of work has investigated signal integrity issues due to process variation and noise [8]–[10]. Researchers mostly improved these attributes considering a power budget as a primary constraint [11]. All of the CDN efforts to improve signal integrity and power are based on traditional voltage-mode (VM) signaling.

As an extension of VM signaling, a wide range of research has been conducted on low-voltage swing signaling [1], differential signaling [12]–[16], pseudo-differential signaling, and incremental signaling [17]. The latter two schemes were only limited to non-clock signal transmission but achieved significant power and performance improvement over full-swing VM schemes.

VM CDNs require clock buffers and the placement of these buffers can disturb timing and require improved clock synthesis methodologies to tackle skew and variability [5], [18]. A current-mode (CM) signal, however, doesn't need distributed buffers and improves the process variation and noise related timing uncertainties [7], [17], [19]. CM signaling has extremely low-voltage swing which enables low dynamic power and also has higher transmission speed compared to its counterpart VM signaling [17], [20]. In addition to power, CM signaling offers superior signal integrity and low switching and substrate noise compared to VM schemes [17].

Recently, attractive circuit techniques for CM clock distribution have been proposed that offer low-power and high signal integrity using current-pulsed flip-flops [3], [7]. However, these schemes were only suitable for symmetric (i.e., equal impedance) clock networks and failed to provide evidence that CM clocking can apply to a real clock network. The primary reason is the lack of existing automation tools to process CM clocks instead of traditional VM clocks. *Balancing insertion delay in a VM clock network is not the same as balancing impedances for CM clocking.* Prior VM algorithms relied on buffers to do this and aren't applicable to CM clocks. In our proposed scheme, we present the first methodology to distribute CM clock signals in real clock networks [21], [22] using a standard-cell design style. Our major contributions are:

- the first clock synthesis methodology to create non-symmetric CM clocks;
- the first demonstration of CM clocking on industrial benchmarks;
- the first standard-cell methodology to utilize CM latch/flip-flop input impedance to minimize skew.

Sections II-III present a brief description of previously reported CM signaling schemes and the motivation of the CM clocking issues. In Section IV, a tuning method is proposed for CM clocks along with a thorough analysis of CM pulsed flip-flop properties and design using them. Section V presents results comparing the proposed CM clocking scheme with existing buffer-based VM CDNs and Section VI concludes the work.

II. Background

Current-mode is widely used for global signaling, especially in high-speed serial links for network buses, memory buses, and multiprocessor interconnection networks [23]. However, at low frequencies, CM signaling consumes large overall power, due to high static power consumption. On the other hand, CMOS logic utilizes VM signaling due to its low static power. CMOS current steering logic has been shown to be robust against digital switching noise, but consumes too much static power [24].

A traditional, point-to-point CM scheme requires a CM transmitter (Tx) and a receiver (Rx) circuit. A Tx circuit

R Islam and M Guthaus are with the Department of Computer Engineering, UCSC, Santa Cruz, CA, 95064 USA e-mail: rislam,mrg@ucsc.edu

This work was supported in part by the National Science Foundation under grant CCF-1053838.

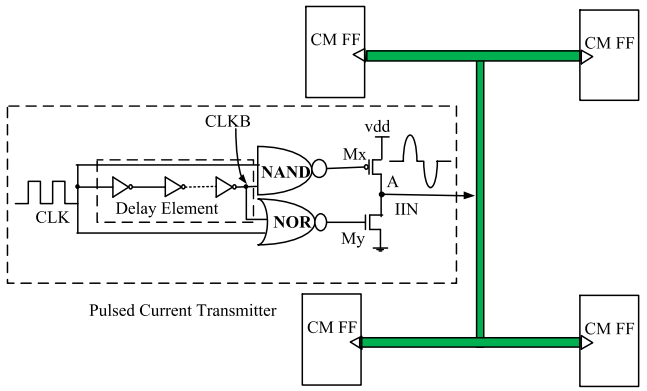

Fig. 1. The previously reported CM clocking scheme saves significant CDN power and exhibits high robustness due to noise and variation, however, only limited to work at symmetric clock networks [7].

ideally converts a VM signal into a CM signal while Rx circuit does the opposite. There have been prior works on these point-to-point networks for both off-chip [25] and on-chip [26] signaling. However, they have not considered point-to-many distribution as needed by clock networks.

One CM clocking scheme for point-to-many clock networks demonstrated significant power and performance improvement over traditional VM clock schemes as shown in Figure 1 [7]. This scheme is based on a low-power CM flip-flop (CM FF) and efficiently applied CM clocking in a hand-designed multi-level H-tree network. The CM-FF-based design used a NAND-NOR Tx that sent a current-pulse converted from a single source VM signal. The Tx generated and transmitted the current-pulse which was synchronized with the rising edge of the input VM clock signal at the Tx. This enabled an edge triggered operation of the Rx circuit in CM FFs. In addition to low power, this scheme showed significant noise robustness compared to the existing VM clocking schemes. However, the work neglected to demonstrate the CM pulsed scheme in a real asymmetric clock network. This needs a new methodology due to CM design issues.

III. Current-Mode Clocking Issues

The trip current of a CM FF is the minimum current to deposit enough charge at a CM FF input so it can store a new value. The clock tree itself remains steady-state at roughly $\frac{V_{dd}}{2}$ and the current pulse arrives nearly instantaneously. Therefore, delay induced skew is not a major issue, unlike VM clocks. In a CM clock, however, an equal amount of current is needed at each FF to prevent timing skew within the CM FF. The main complication is that the duration and peak, and hence total charge, of the current pulse must be within bounds.

Balancing the impedance at each wire branch is not a trivial task because it depends on the input impedance of the FF inputs. Prior VM methods could decouple downstream impedance using buffers but CM has an advantage in performance and power by not using buffers. In addition, the Tx at the root determines the steady-state voltage of the clock network which defines the bias point of the FF clock input.

The FF input impedance changes depending on the input current and the bias point set by the Tx, which effectively means that the CM FF changes input impedance during a typical clock pulse when there are slight bias fluctuations. The current steered at each branching point depends on each branch's impedance but this, in turn, depends on the downstream FFs and the current that is steered to them. Because of this challenge, previous CM clocking has been restricted to symmetric H-trees [7], [26].

As a result of trip current mismatch, the internal CM FF voltage pulse (CLKP) can vary in the time-domain and result in clock skew. This inaccuracy can increase quickly in larger asymmetric networks with large variation in current at the sinks. In the worst case, a CM FF may not respond if the trip current is insufficient which can result in a functional failure. Hence, it is desirable to use an automated synthesis tool not only for automation of the routing and impedance balancing, but also to ensure the electrical correctness and functionality.

VM clock synthesis techniques typically use Elmore delay models for initial clock routing and then insert and balance buffers to constrain the network's slew rates. Since the Elmore delay model is based on the charging/discharging of a capacitance through a resistance, it is not suitable for CM synthesis because CM clocking maintains a steady-state voltage in the entire clock network. Elmore-delay-based clock routing balances delays in clock branches which is not the same as balancing impedances. However, it is a reasonable starting point and can be compensated for by appropriately sizing the Tx and the Rx circuitry in the CM FF.

To demonstrate the skew improvement after proposed Tx sizing and CM FF sizing stages, we performed synthesis and simulation of different routing techniques in Figure 2 on an four sink, asymmetric CM clock distribution using the previously reported CM Tx and FF circuits [7]. Since a symmetric H-tree network doesn't work well with asymmetric distributions, it routes to a fixed location depending on the size of the H-tree. This results in a large $19.1ps$ skew as shown in Figure 2(a) [7], [26]. Using a deferred merge-embedding (DME) methodology and CM clocking, we observed a better, but, still considerable $14.8ps$ skew as shown in Figure 2(b). The skew improvement is due to the balanced RC product in each sub-tree. Using our proposed iterative Tx sizing methodology with a DME tree, we observe improvement to $3.1ps$ skew as shown in Figure 2(c). Sizing the Rx in the CM FF further improves the impedance matching and compensates for skew using the clock-to-internal voltage pulse (CLK-CLKP) delay of the CM FF. Using this technique along with the Tx sizing, the skew is $1.6ps$ as shown in Figure 2(d). However, this is a small four sink motivational example that is intended to illustrate the principle of the work. It is not meant to be verification of the methodology which is reserved for the benchmark results in Section V.

In addition to skew, it is expected to have lower-jitter induced timing uncertainty in CM clocking compared to a VM scheme due to the absence of buffers in CM CDN and jitter will not be addressed further in this work.

Our research provides an automated methodology for the Tx and CM FF Rx sizing. It is worth mentioning that the proposed methodology is in stark contrast to the existing impedance balancing VM schemes [8], [27] where clustering

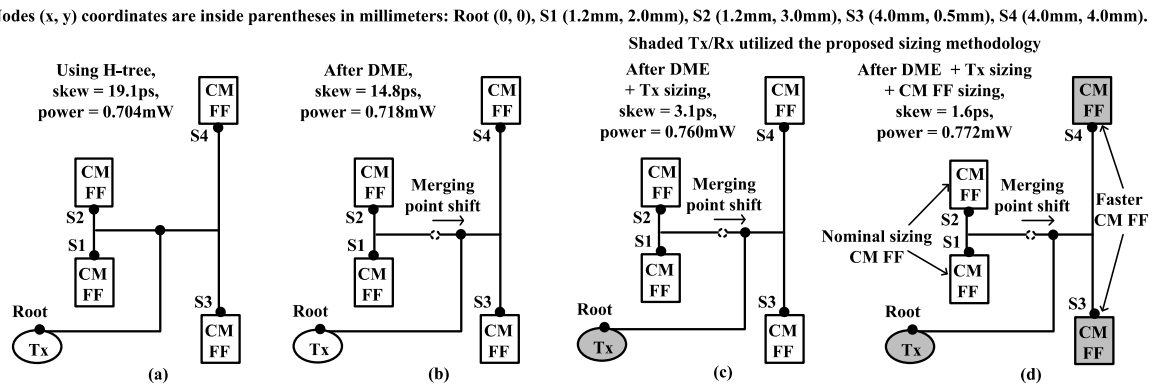

Fig. 2. Both symmetric and DME VM synthesis techniques introduce large skews ($19.1ps$ and $14.8ps$, respectively) when directly applied to asymmetric CM clock distributions, however, DME with Tx or combined Tx/Rx sizing methodology can improve the clock skew to $3.1ps$ and $1.6ps$, respectively, with almost equal power consumption in each case.

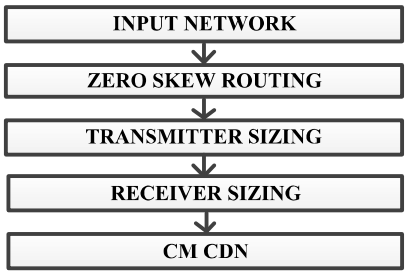

Fig. 3. The flowchart of the proposed CMCS scheme uses a zero-skew unbuffered clock routing along with stages to set the bias voltage with Tx sizing and Rx sizing to minimize skew and maintain correct functionality.

and load balancing was achieved using wire and/or buffer sizing [27]. Even timing model independent schemes utilized extra wires and dummy sinks to balance the network [8], but these schemes are only suitable for buffered VM clocking, since the CM FF also have varying input impedance.

IV. Proposed Current-Mode Clock Synthesis (CMCS)

The reliability and overall performance of a CM clocking scheme depends greatly on the Tx and Rx/CM FF circuits and their transistor sizes. The advantage, however, is a tremendous amount of power savings with similar skews compared to existing buffered VM clocking methodologies.

The overview of proposed CMCS scheme is shown in Figure 3 which starts with a traditional DME tree construction. While this is not exactly optimal for impedance matching, it generally is a good starting point. It is followed by a stage of Tx sizing to determine the appropriate bias voltage of the network and then an iterative skew improvement through Rx sizing in the CM FFs.

A. CM Pulsed Current Transmitter Sizing

The proposed CM clock networks are unbuffered and driven at the root by a CM Tx [7]. The CM Tx generates a push/pull current and the devices are sized so that the network maintains a steady-state bias voltage. Since the Tx is large, it may have several exponentially tapered stages of buffers driving it, which are included in our later results. The detailed algorithm for our CM pulsed current Tx sizing is presented in Algorithm 1.

We performed a wide range of simulations on different size and topology networks to relate the Tx sizing with the total capacitive admittance (Y_T) of the network. The result of these experiments are shown in Figure 4. The relationship is highly linear between the Y_T and the Tx size.

In order to relate the total driving load/impedance with the Tx size, we calculate the total impedance of the network. However, it is tradition to use admittance, which is simply the inverse of impedance, for parallel networks. The total admittance of a network is proportional to the current as shown in Figure 4. We calculate the total admittance of a CDN by considering the total FF load and the RC network. The input admittance of a CM FF is

$$Y_{in} = g_{m1} + g_{m2} = C_{ox} \cdot AR \cdot V_{OV} \cdot (\mu n + \mu p) = \alpha C_{ox} \quad (1)$$

where g_{m1}, g_{m2} are the transconductance of the receiving transistors, μn, μp are the mobility of NMOS and PMOS transistors, and C_{ox} is the gate oxide capacitance. The Aspect-Ratio (AR = W/L = width/length) of Mr1-Mr2 in Figure 5 determine the input admittance. V_{OV} is the overdrive voltage of transistor which depends on the bias point. This equation can be simplified using a variable α and assuming all the capacitance in the CDN are in parallel (connected from $\frac{V_{dd}}{2}$ to ground). Now we can write the Y_T of an entire clock network with the FFs as

$$Y_T = \beta\Big(\sum_{i \in sinks} \alpha_i C_{ox} + \sum_{j \in wires} C_{w,j}\Big) \quad (2)$$

where $C_{w,j}$ is the wire capacitance of wire j, α_i is the admittance factor of sink i and β is a constant. We can utilize the linearity of Y_T and Tx size to parameter fit β as a starting point. The error bounds suggestion that a $\pm 12\%$ range around the starting point should be considered during optimization. The α_i values are optimized later in Section V-B when we select CM FF library cells with varying AR sizes. The first part of the Equation 2 ensures the total required current at each sink while the latter part helps the Tx to sustain $\frac{V_{dd}}{2}$ voltage and the fraction of energy loss due to non-ideal voltage swing on the interconnect.

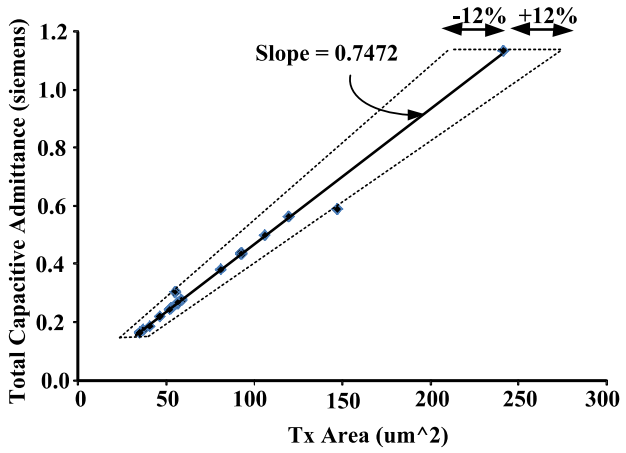

Fig. 4. Ideal CM Tx sizing varies linearly with the total capacitive admittance of the clock network which allows linear fitting for a starting Tx size.

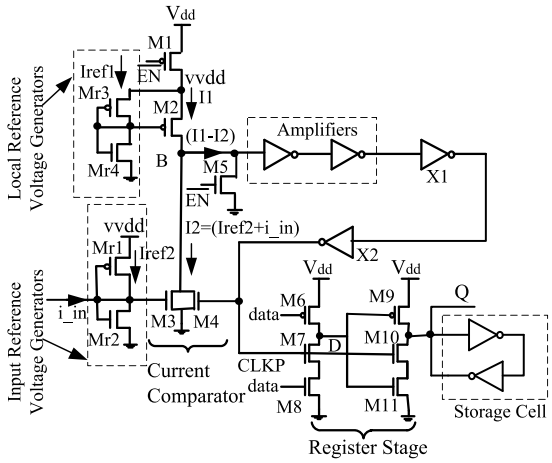

Fig. 5. Sizing of CM FF reference-voltage generators changes the FF internal CLK-CLKP time resulting in faster or slower FF with no impact on FF timing constraint [7].

Algorithm 1 Current transmitter sizing

1: **Input:** Zero skew routed tree (Tree);
2: **Output:** Properly sized transmitter
3:
4: $Y_T = totalAdmittance(Tree)$
5: $T_{init} = sizeTransmitter(Y_T)$
6: $simulateTransient()$
7: $S_{init} = calculateSkew()$
8: $S_{best} = S_{init}, T_{best} = T_{newUp} = T_{newDown} = T_{init}$
9: **while** $S_{new} \leq S_{best}$ **do** ▷ repeat if improvement or equal
10: $T_{newUp} = T_{newUp} + \delta s$ ▷ δs is the 1% of T_{init}, sizing up
11: $simulateTransient()$
12: $S_{new} = calculateSkew()$
13: **if** $S_{new} < S_{best}$ **then**
14: $S_{best} = S_{new}$, $T_{best} = T_{newUp}$
15: **end if**
16: **end while**
17: **while** $S_{new} \leq S_{best}$ **do** ▷ repeat if improvement or equal
18: $T_{newDown} = T_{newDown} - \delta s$ ▷ sizing down
19: $simulateTransient()$
20: $S_{new} = calculateSkew()$
21: **if** $S_{new} < S_{best}$ **then**
22: $S_{best} = S_{new}$, $T_{best} = T_{newDown}$
23: **end if**
24: **end while**

Empirically the Tx sizing is convex, so we used steepest descent search to find the best size. The Tx sizing algorithm first calculates the Y_T of the network (Line 4) in the $totalAdmittance(Tree)$ method which applies Equation 2. Then it determines the initial Tx sizing (T_{init}) of the network (Line 5) using $sizeTransmitter(Y_T)$. It runs a transient simulation ($simulateTransient()$) and uses $calculateSkew()$ to measure the initial skew (S_{init}) (Lines 6-7). T_{best} and S_{best} are set to the initial values (T_{init} and S_{init}), respectively (Line 8). The T_{init} value is also stored in two temporary variables (T_{newUp} and $T_{newDown}$).

After this, the algorithm sweeps up and down from T_{init} with a step size of δs which is assumed to be 1% of T_{init} using two independent loops (Lines 9-24). The change in Tx device sizes also changes the network bias voltage and the input current of a CM FF that effectively changes the CLK-CLKP delay of the FF in Figure 5. In addition, the DME based tree does not guarantee equal impedance of each branch resulting CLK-CLKP delay mismatch. This can change the skew of the network and it is imperative to calculate the new skew with the resized Tx. During each iteration, the Algorithm compares the new simulated skew (S_{new}) with the previous best skew and retains the best skew (S_{best}) along with corresponding Tx size (T_{best}). The Algorithm terminates if there is no improvement in skew. This proposed Tx sizing methodology has worked with any network and our experimental results in Section V-C will show the quality.

B. Receiver/CM FF sizing Methodology

To aid skew optimization, we utilize a small set of pre-designed CM FF library cells with different input impedances. The input impedance is changed by varying the AR of the input reference voltage generator (Mr1-Mr2) diode-connected inverter circuits in Figure 5 as modeled in Equation 1. However, it is necessary to have equal AR for both the input reference voltage generator and local reference voltage generator (Mr3-Mr4) to measure the correct trip current of a CM FF. Because of that we change the AR of both voltage generators simultaneously. This results in a voltage variation at the input of the Current-Comparator (CC) and can move the bias-point. The variation of bias voltage also varies the CLK-CLKP delay of CM FF. These results are shown later in Section V-B.

The proposed CM FF sizing methodology balances the root to sink admittance of an unbalanced tree by selecting among the available CM FF library cells. Since these cells have different admittance, they have differing internal CLK-CLKP delays which can be used to balance any skew. We approach the CM FF sizing problem by starting with a median CLK-CLKP delay FF and replacing those that have lower or higher impedance (with faster or slower versions), respectively.

The detailed algorithm for our sizing is shown in Algorithm 2. The FFs are initially set to the median size to allow

Algorithm 2 CM Pulsed FF sizing

1: **Input:** Zero skew routed tree (Tree) without buffers; properly sized Tx; a set of CMPFFE library cells (L); skew bound (SB)
2: **Output:** Properly sized CM pulsed FFs that meets the skew requirement
3:
4: $simulateTransient()$
5: $S_{init} = calculateSkew()$
6: $S_{best} = S_{init}$
7: **while** $S_{best} > SB$ **do**
8: $(critSinks, goodSinks) = findCriticalSinks()$
9: $d_{max} = maxDelay(goodSinks)$
10: $d_{min} = minDelay(goodSinks)$
11: **for all** $s_i \in critSinks | d_{s_i} < d_{min}$ **do**
12: $slowerFlipflop(s_i)$ ▷ replace with slower FFs
13: **end for**
14: **for all** $s_i \in critSinks | d_{s_i} > d_{max}$ **do**
15: $fasterFlipflop(s_i)$ ▷ replace with faster FFs
16: **end for**
17: $simulateTransient()$
18: $S_{new} = calculateSkew()$
19: **if** $S_{new} < S_{best}$ **then**
20: $S_{best} = S_{new}$
21: **else**
22: $break$ ▷ terminate if no improvement
23: **end if**
24: **end while**

them to be made faster/slower. After a transient simulation, the algorithm calculates the S_{init} (Line 4-5) and sets S_{best} as S_{init} (Line 6). We search over the sinks' timing information and determine the set of sinks that need improvement in $findCriticalSinks()$ (Line 8). Then, the algorithm iteratively resizes the critical CM FFs until we meet our skew bound (SB) (Lines 7-24).

The $findCriticalmethod()$ function identifies the largest cluster of FFs in any skew bound window as the "good" sinks. Algorithm 3 does this by iterating over a list of sinks sorted by their delay (D_{in}) (Line 4) and counting the number of sink delays d_j within a skew bound (SB) from sink i with delay d_i (Lines 8-13). The largest number of sinks in a window ensures that the fewest CM FFs will be returned in the critical sink set C and need to be adjusted in Algorithm 2. These "critical" sinks are outside the optimal window can be either too fast or too slow.

Algorithm 3 has a worst case runtime complexity of $O(n^2)$, where n is the number of sinks. However, the SB is small and we only look into the set of sinks within a skew bound, which severely limits the second n. This makes the proposed Algorithm linear in practice. In addition, using linear time maximal sum Algorithm [28], the proposed Algorithm 3 could be speed-up to $O(n)$. However, the runtime is dominated by simulation and not the algorithm itself so we did not do this.

During each iteration of Algorithm 2, we calculate the maximum delay (d_{max}) and minimum delay (d_{min}) of the "good" sinks (Line 9-10). Then two consecutive loops iterate over the fast and slow critical sinks, respectively, and choose a faster/slower CM FF from the library cells (Lines 11-16). A transient simulation calculates the new skew (S_{new}) and stores the minimum value to S_{best} after comparison (Lines 17-23).

The proposed CM FF sizing algorithm converges to a minimum skew after either no skew improvement is seen or the skew bound is achieved. It is worth mentioning that the CM FF are sized to meet the SB for a fixed Tx size, which was determined in the previous stage. The Tx is not sized after the receivers. So there is no need to size the Tx again. In addition, the CM FFs are very fast and Algorithm 1 ensures proper functionality of each FF by properly sizing the CM pulsed current Tx. FF metastability is usually due to the input arriving during a clock transition. Our CM FF still has setup and hold times like VM FFs to avoid any such problems.

Algorithm 3 Finding critical sinks

1: **Input:** Time delay of each sink (D_{in}); set of sinks (S); skew bound (SB)
2: **Output:** Set of critical sinks (C) beyond the skew bound and good sinks set (G)
3:
4: $D = sort(D_{in})$ ▷ sort all the sinks based on their time delay
5: $index_{best} = 0$, $cnt_{best} = 0$, $i = 0$
6: **while** $i++ < |D|$ **do**
7: $j = i + 1$, $cnt = 0$
8: **while** $d_j < (d_i + SB) | (d_i, d_j) \in D$ **do**
9: $cnt++$, $j++$
10: **if** $cnt > cnt_{best}$ **then**
11: $index_{best} = i$, $cnt_{best} = cnt$
12: **end if**
13: **end while**
14: **end while**
15: $G = s | d_{index_{best}} \leq d_i \leq d_{index_{best}+cnt_{best}}$ ▷ store all the good sinks in G
16: $C = S \backslash s$ ▷ store all the critical sinks in C

V. Simulation Results

A. Simulation Setup

We implemented the proposed CMCS scheme in C++ and Python. Simulations were run on an Intel Core i5-3570 Ivy Bridge 3.4GHz quad-core processor. We validate the proposed methodology using 45nm ISPD 2009 and 2010 industrial Benchmarks [21], [22]. ISPD 2009 benchmarks are derived from real IBM ASIC designs. These benchmark circuits are distributed in $50.4 - 275.6mm^2$ area and consists of 81-623 evenly/unevenly distributed sinks with equal or unequal sink capacitances. ISPD 2010 benchmarks are derived from real IBM and Intel Microprocessor designs. The 2010 benchmark circuits are distributed in $1.4 - 91.0mm^2$ area and consists of 981-2249 nonuniformly distributed sinks with different loading. Our designs were optimized for 1V supply voltage and clock frequencies range from 1-3GHz. Traditionally, 5-10% of the clock period is allocated for clock skew, so we used a clock skew bound of 70 ps for 1 GHz clock frequency. Traditionally worst case slew rate is defined as 10% of the clock period. For the proposed CM clocking schemes, we used 10% slew bound. It is worth mentioning that at steady state the CM clock tree remain roughly around $\frac{V_{dd}}{2}$, hence we only considered worst case slew rate at the CLKP signal of CM FF. The CM Tx and Rx/FF [7] were designed using the FreePDK 45nm CMOS technology [29]. We used HSPICE to measure power and performance for all results.

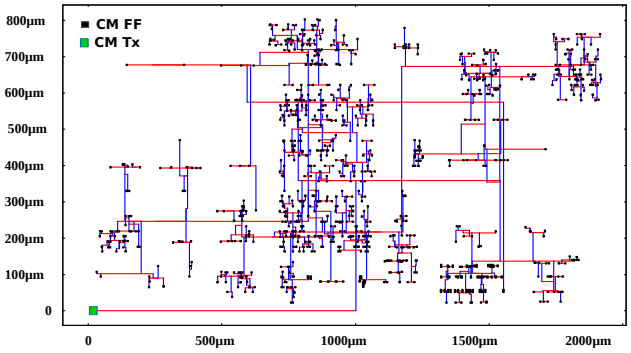

Fig. 6. The resulting routed CM CDN for the ISPD 2010 benchmark circuit 06.

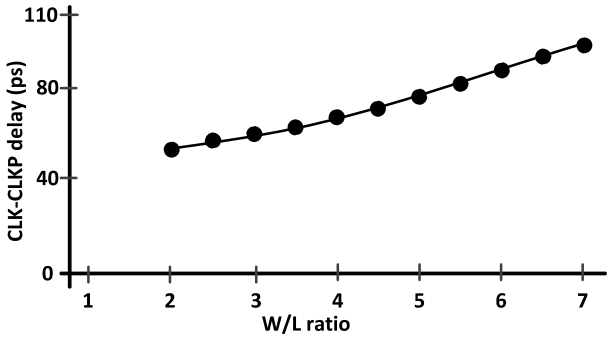

Fig. 7. CM FF library cells are built based on the characteristics that the CM pulsed FF CLK-CLKP delay increases with the increase of aspect-ratio ($\frac{W}{L}$).

The clock tree is routed with minimum wire length by incorporating balanced bipartition (BB) with DME [9], [10] and the final tree nodes are connected to the CM FFs. The clock tree and the CM FFs are driven by a single pulsed current Tx. In addition, we followed ISPD 2010 High Performance Clock Network Synthesis Contest guideline to model the clock network as a distributed RC model [21], [22]. The CM Tx, tree, and the CM FFs compose the entire CM network. Figure 6 shows the resulting DME routed bufferless CM CDN for the ISPD 2010 Benchmark circuit 06. In proposed CMCS scheme, the total power consumption includes the CM pulsed Tx power, parasitic power, and the total CM FF power.

The CMCS methodology uses library cells of CM FF with different AR and hence input impedance and CLK-CLKP delay resulting "slower" and "faster" FFs. Here "faster" and "slower" FF refers to the smaller and larger CLK-CLKP delays, respectively. We calculate global clock skew at the FF's internal clock pins (CLKP), so that changes in CLK-CLKP delay are included in the skew component of timing constraints and do not change the setup time and hold time.

It would be interesting to compare the CMCS results with the ISPD 2009 and ISPD 2010 winners results. But, the winning teams consider local skew minimization resulting in wire snaking and extra buffers. For example, using the 01 benchmark circuit the ISPD 2010 winning team used $198.3pF$ capacitance, while the implemented VM network requires $93.7pF$ capacitance. Overall, ISPD 2009 and ISPD 2010 winners consume significantly more capacitance resulting more than double power consumption compared to our implemented buffered VM networks, hence in our final comparison we eliminated ISPD winners result.

Since the previous Tx sizing methodology [7] does not work with asymmetric networks, we used a state-of-the-art buffered VM methodology for comparison. The VM tree is routed using a common industry method with minimum wire length [9], [10] and the buffers are inserted to meet the skew and slew constraints (10% of the clock period) [30]. For the VM buffered network, the total power consumption includes CDN buffer power, clock tree parasitic power, and VM pulsed FF [31] power. Both the VM and CM schemes receive a traditional voltage clock from a PLL/CLK divider at the root. The input CLK signal slew rate is 10% of the CLK period.

B. CM FF Library Cells

Similar to a VM FF, in the CM case we considered 50% ideal input current ($3\mu A$) transition to 50% Q transition as the CLK-to-Q delay of CM FF. For setup (t_s) and hold time (t_h times we used the common definition as the time margin that causes a CLK-to-Q delay increase of 10% beyond nominal. The t_s and t_h of the median size CM FF are $-15.8ps$ and $46.6ps$, respectively. Figure 7 shows an analysis of the CM FF library cells with the nominal input current of $\pm 3\mu A$ and $70ps$ pulse width. In this analysis, we vary the AR of CM FF reference voltage generators and measure the corresponding CLK-CLKP delay. We observed a linear relationship between CLK-CLKP delay with AR. Particularly, the CLK-CLKP delay of the CM FF increases with the increase of AR by increasing input impedance as shown in Equation 1. Hence, we utilized this characteristic to build our CM FF library cells with different CLK-CLKP delay. It is worth mentioning that, similar to a FF output (Q) signal, the CLKP act as both terminal and voltage pulse.

In order to tackle skew issues, the proposed CMCS utilized 13 CM FF library cells (a median size and 6 faster and 6 slower) with $\pm 30ps$ CLK-CLKP delay variation from the nominal delay value. It is expected that the use of different sizing CM FF requires different FF area and may add area overhead to the overall design. However, It is possible to have zero area overhead for different size FF. Figure 8 shows the layout of fastest, median, and slowest CLK-CLKP delay CM FF. In Figure 8, the P_n and N_n indicates the sizing reference of PMOS and NMOS, respectively; corresponding to reference voltage generator of median size CM FF. We laid out the CM FF in such a way that we can adjust the sizing of CM FF reference voltage generator without changing the CM FF overall area. Since, each FF used standard cell height, we can adjust the AR by using vertical empty space for slower CM FF (larger transistors) or decrease transistors size in the opposite direction (for faster CM FF) as shown in Figure 8 (c) and Figure 8 (a), respectively. This requires no placement legalization.

We characterized the register stage of each CM FF considering maximum driving load. In addition, the CLKP signal has

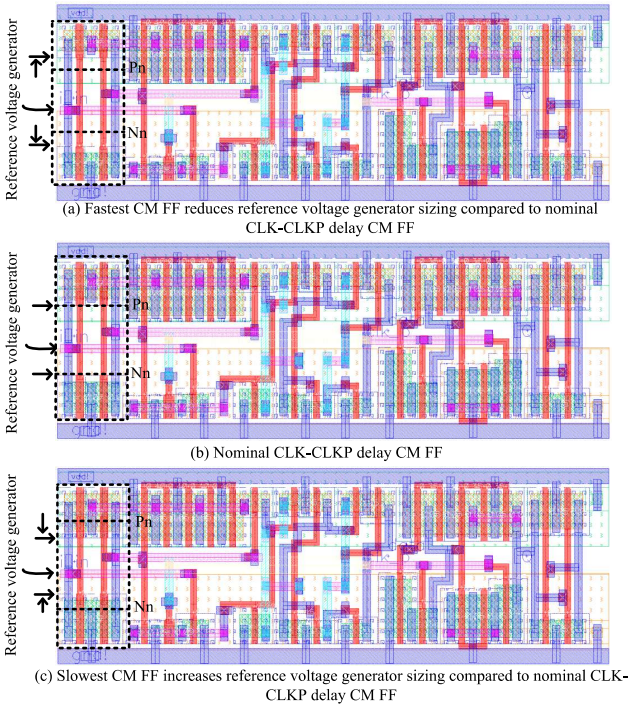

Fig. 8. The CM FF library cells (fastest, median, and slowest CM FFs) consumes same area, resulting zero area overhead due to FF sizing methodology.

fixed loading from transistors M4, M7, and M10 as shown in Figure 5. If the CLKP signal meets a slew rate, there is no slew rate violation at the CM FF output (Q) signal.

C. Results and Comparisons

Table I shows the power, skew, run time comparison on the ISPD 2009 benchmarks while Table II shows the ISPD 2010 benchmark networks. We extracted all the results considering the final Tx and CM FF sizes for CM networks.

1) Power Comparison: Table I and Table II show the power breakdown of the VM and CM FFs and total CDN simulation power at 1 GHz clock frequency. At 1GHz the CM FFs consume 24% and 20% more average power than VM FFs using ISPD 2009 and ISPD 2010 testbenches, respectively. On the other hand, the CM Tx consumes 97% and 92% lower average power compared to VM buffers power on ISPD 2009 and ISPD 2010 networks, respectively. This is due to the full voltage swing ($0 \rightarrow V_{dd}$) in the VM CDN, whereas the CM CDN has negligible voltage swing. Overall, using proposed CMCS methodology, the CM clocking consumes lower power than the traditional buffered VM clocking on all the ISPD 2009 and 2010 benchmarks. Specifically, the CM clocks save 68-90% power compared to the VM buffered networks as shown in Table I and up to 67% power in Table II.

In a CM scheme, most of the power is static power consumed by the CM FFs and there are no CDN buffers so it is highly insensitive to frequency [7]. Because of this, CM clocking save quadratically more power at higher frequencies which is extremely important in multi-GHz designs. Figure 9 shows the evidence of the proposed CMCS methodology efficiency compared with VM buffered scheme at higher frequencies using ISPD 2009 benchmark circuit s4r3. In particular, the power saving of CM methodology increases from 68% (at 1GHz) to 84% (at 3GHz) compared to VM scheme.

2) Skew Comparison: The proposed algorithm reduces skew by Tx and CM FF sizing while ensuring correct functionality. The CMCS methodology resulted in proper functionality in all of the asymmetric networks. The skew slightly degraded on average in both the 2009 and 2010 benchmarks, but the skew results were better on some benchmarks as shown in Table I and Table II. These skew levels are well within tolerable limits of 5-10% of the clock period and are therefore not a concern especially considering the large power consumption savings. In addition, each scheme uses a different methodology the response to optimization is not predictable. This is common with any sort of heuristic optimization algorithm which may end up in a solution that is closer or further from optimal. However, overall the proposed CM scheme has only $3.3ps$ and $3.9ps$ average skew difference compared to VM scheme for ISPD 2009 and ISPD 2010 testbenches, respectively.

3) Run-Time Comparison: Most high-performance CDNs use HSPICE simulation instead of approximate analytical models such as Elmore delay in traditional clock tree synthesis (CTS) algorithms. However, HSPICE simulation requires significant simulation time compared to a traditional CTS algorithm. Table I and Table II show the results based on accurate HSPICE simulation for both VM and CM methodologies for fair comparison of quality of results and run-time.

The run time of the CMCS methodology is significantly less than the VM methodology. This is because, the proposed scheme requires fewer iterations since it doesn't use buffers that need to be sized. Overall, the run time of the benchmarks are $2.4-9.1\times$ less on average as shown in Table I and Table II.

4) Silicon Area Comparison: Similar to previous CM clocking systems, the proposed CMCS scheme uses a bufferless CDN. However, the Tx circuit has a few buffers for

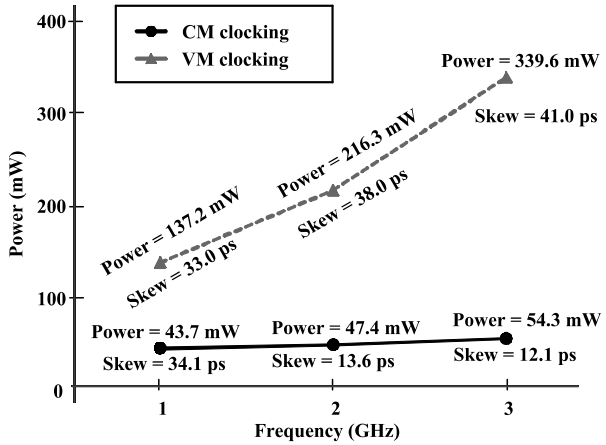

Fig. 9. The CM clocking is highly insensitive to frequency, as a result it exhibits more power saving at higher frequencies; for example using ISPD 2009 benchmark s4r3 circuit, the power saving of CM methodology increases from 68% (at 1GHz) to 84% (at 3GHz) compared to VM scheme.

TABLE I

THE PROPOSED CMCS SCHEME ENABLES MORE THAN 82% AVERAGE POWER SAVING USING CM CLOCKING AND 9× LOWER AVERAGE RUN-TIME(CPU) WITH ONLY 3.3ps SKEW DEGRADATION COMPARED TO THE TRADITIONAL VM BUFFERED SCHEME USING 2009 ISPD BENCHMARKS

Benchmark				VM Buffered network				CM network				**CM compared to VM**		
Name	Sink	Chip area	Total cap	Power (mW)		Skew	Run time	Power (mW)		Skew	Run time	Power	Skew	Run time
	(#)	(mm^2)	(pF)	VM FF	Total	(ps)	(hr)	CM FF	Total	(ps)	(hr)	(%)	(ps)	(VM/CM)
s1r1	81	69.4	27.6	3.7	38.5	14.0	0.31	4.9	6.0	20.9	0.11	84.5	-6.9	2.8
s2r1	88	54.6	26.3	4.0	37.9	20.0	0.50	5.3	6.5	21.1	0.12	82.9	-1.1	4.1
s3r1	131	165.6	48.4	5.9	69.4	30.0	1.60	7.9	9.7	67.8	0.42	86.1	-37.8	3.8
s4r3	623	120.7	79.5	28.0	137.2	33.0	4.95	37.4	43.7	34.1	0.62	68.2	-1.1	8.0
f11	121	109.2	42.2	5.4	60.2	14.0	0.53	7.3	9.2	16.7	0.06	84.7	-2.7	9.0
f12	117	91.2	39.5	5.3	57.4	20.0	1.43	7.0	8.5	28.3	0.14	85.1	-8.3	10.5
f21	117	133.3	43.9	5.3	62.0	28.0	0.54	7.0	8.6	18.5	0.06	86.1	9.5	9.2
f22	91	50.4	26.1	4.1	37.8	12.0	0.27	5.5	6.7	19.1	0.12	82.2	-7.1	2.3
f31	273	275.6	89.7	12.3	131.7	37.0	3.61	16.5	19.8	19.1	0.22	85.0	17.9	16.6
f32	190	269.0	69.5	8.6	100.3	23.0	6.03	8.7	10.3	18.3	0.31	89.7	4.7	19.6
Avg.	183	133.9	49.3	8.2	73.2	23.1	1.98	10.8	12.9	26.4	0.22	**82.4**	**-3.3**	**9.1**

TABLE II

USING MORE DENSE CLOCK SINKS ISPD 2010 BENCHMARKS THE CMCS SCHEME CONSUMES 39% LOWER AVERAGE POWER AND 2.4× LOWER AVERAGE RUN-TIME(CPU) HOWEVER, EXPERIENCED 3.9ps SKEW DEGRADATION COMPARED TO THE VM SCHEME

Benchmark				VM Buffered network				CM network				**CM compared to VM**		
Name	Sink	Chip area	Total cap	Power (mW)		Skew	Run time	Power (mW)		Skew	Run time	Power	Skew	Run time
	(#)	(mm^2)	(pF)	VM FF	Total	(ps)	(hr)	CM FF	Total	(ps)	(hr)	(%)	(ps)	(VM/CM)
01	1107	64.0	93.7	49.8	157.5	32.0	8.0	55.9	60.5	42.7	1.58	61.6	-10.7	5.1
02	2249	91.0	180.4	101.2	305.8	32.0	11.0	118.5	128.3	20.2	4.02	58.0	11.8	2.7
03	1200	1.4	42.5	54.0	90.8	33.0	2.5	67.5	71.3	23.2	2.03	21.5	9.8	1.2
04	1845	5.7	69.5	83.0	128.1	33.0	5.0	116.2	126.3	53.2	2.46	1.5	-20.2	2.0
05	1016	5.8	29.6	45.7	67.0	26.0	2.0	48.4	50.4	25.8	1.38	24.8	0.2	1.4
06	981	1.5	34.9	44.1	147.5	22.0	4.0	46.6	48.5	37.1	1.32	67.1	-15.1	3.0
07	1915	3.5	60.7	86.2	132.8	30.0	4.2	117.4	129.0	39.4	2.47	2.9	-9.4	1.7
08	1134	2.6	38.9	51.0	92.0	32.0	3.9	69.5	75.5	29.9	1.32	18.0	2.1	3.0
Avg.	1431	21.9	68.8	64.4	140.2	30.0	5.1	80.0	86.2	33.9	2.07	**38.5**	**-3.9**	**2.4**

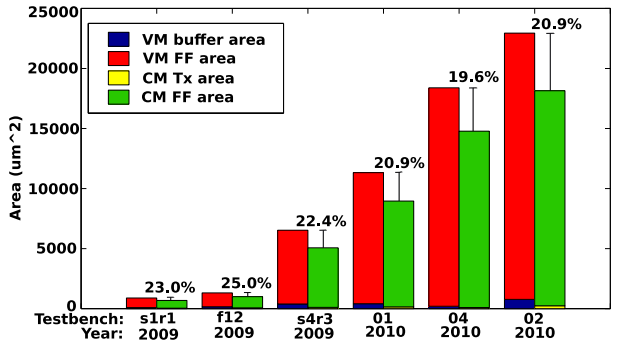

Fig. 10. The proposed Algorithm saves 20% to 25% silicon area as a result of bufferless clock routing using ISPD 2009 and ISPD 2010 benchmarks.

the internal delay chain and to drive the large Tx transistors. Figure 10 shows a representative comparison of VM buffered total area compared to CM total area. The CM CDN includes the overhead of the resized FFs and Tx to compute the Tx and CM FF area. When considers CM Tx and VM buffers area, the CM clocking saves up to 73% transistor area compared to the VM scheme. Overall, using proposed CMCS methodology in ISPD 2009 and ISPD 2010 benchmarks, the CM clocking saves 21% average silicon area compared to VM scheme.

VI. CONCLUSION

We have presented the first current-mode clock synthesis (CMCS) methodology. The proposed methodology used transmitter (Tx) and receiver (Rx) sizing in the CM flip-flops to ensure correct functionality and reduce skew. The proposed methodology saved $39-84\%$ average power with similar skews on industrial benchmarks. In addition, the methodology used $2.4-9.1\times$ less run-time up to 26% lower silicon area compared to the buffered VM networks.

REFERENCES

[1] Q. K. Zhu, and M. Zhang, "Low-voltage swing clock distribution schemes," *ISCAS*, May 2001, pp. 418–421.

[2] M. A. El-Moursy, and E. G. Friedman, "Exponentially tapered H-tree clock distribution networks," *TVLSI*, vol. 13, no. 8, pp. 971–975, Aug 2005.

[3] R. Islam and M. R. Guthaus, "Current-mode clock distribution," *ISCAS*, Jun 2014, pp. 1203–1206.

[4] I. Bezzam, C. Mathiazhagan, T. Raja, and S. Krishnan, "An Energy-Recovering Reconfigurable Series Resonant Clocking Scheme for Wide Frequency Operation," *TCASI*, vol. 62, no. 7, pp. 1766–1775, July 2015.

[5] D. Papa, C. Alpert, C. Sze, Z. Li, N. Viswanathan, Gi-Joon Nam, and I. L. Markov, "Physical Synthesis with Clock-Network Optimization for Large Systems on Chips," *IEEE Micro*, vol. 31, no. 4, pp. 51–62, Jul 2011.

[6] S. Chen, H. Li, and P. Y. Chiang, "A Robust Energy/Area-Efficient Forwarded-Clock Receiver With All-Digital Clock and Data Recovery in 28-nm CMOS for High-Density Interconnects," *TVLSI*, vol. 24, no. 2, pp. 578–586, Feb 2016.

[7] R. Islam and M. R. Guthaus, "Low-Power Clock Distribution Using a Current-Pulsed Clocked Flip-Flop," *TCASI*, vol. 62, no. 4, pp. 1156–1164, Apr 2015.

[8] Xin-Wei Shih and Yao-Wen Chang, "Fast timing-model independent buffered clock-tree synthesis," *DAC*, Jun 2010, pp. 80–85.

[9] R.-S. Tsay, "Exact zero skew," *ICCAD*, Nov 1991, pp. 336–339.

[10] K. D. Boese and A. B. Kahng, "Zero-skew clock routing trees with minimum wirelength," *ASIC*, Sep 1992, pp. 17–21.

[11] M. R. Guthaus, G. Wilke, and R. Reis, "Revisiting automated physical synthesis of high-performance clock networks," *TODAES*, vol. 18, no. 2, pp. 31:1–31:27, Apr 2012.

[12] R. Islam, H. Fahmy, Ping-Yao Lin, and M. R. Guthaus, "Differential current-mode clock distribution," *MWSCAS*, Aug 2015, pp. 1–4.

[13] S. C. Chan, K. L. Shepard, and P.J. Restle, "Distributed Differential Oscillators for Global Clock Networks," *JSSC*, vol. 41, no. 9, pp. 2083–2094, Sept 2006.

[14] A. Narasimhan, M. Kasotiya, and R. Sridhar, "A low-swing differential signalling scheme for on-chip global interconnects," *ICVD*, Jan 2005, pp. 634–639.

[15] A. Maheshwari and W. Burleson, "Differential current-sensing for on-chip interconnects," *TVLSI*, vol. 12, no. 12, pp. 1321–1329, Dec 2004.

[16] D. C. Sekar, "Clock trees: differential or single ended?," *ISQED*, Mar 2005, pp. 548–553.

[17] Fei Yuan, "CMOS Current-Mode Circuits for Data Communications," *Springer*, Apr 2007.

[18] Z. Li, C. J. Alpert, Gi-Joon Nam, C. Sze, N. Viswanathan, and N. Y. Zhou, "Guiding a physical design closure system to produce easier-to-route designs with more predictable timing," *DAC*, Jun 2012, pp. 465–470.

[19] N. K. Kancharapu, M. Dave, V. Masimukkula, M. S. Baghini, and D. K. Sharma, "A Low-Power Low-Skew Current-Mode Clock Distribution Network in 90nm CMOS Technology," *ISVLSI*, Jul 2011, pp. 132–137.

[20] A. P Jose, G. Patounakis, and K.L. Shepard, "Near speed-of-light on-chip interconnects using pulsed current-mode signalling," *VLSIC*, Jun 2005, pp. 108–111.

[21] C. N. Sze, P. Restle, G. J. Nam, and C. J. Alpert, "Clocking and the ISPD'09 clock synthesis contest," *ISPD*, Mar 2009, pp. 149–150.

[22] C. N. Sze, "ISPD 2010 High Performance Clock Network Synthesis Contest," *ISPD*, Mar 2010.

[23] K. Farzan, and D. A. Johns, "A CMOS 10-gb/s power-efficient 4-PAM transmitter," *JSSC*, vol. 39, no. 3, pp. 529–532, Mar 2004.

[24] Hiok-Tiaq Ng and D. J. Allstot, "CMOS current steering logic for low-voltage mixed-signal integrated circuits," *TVLSI*, vol. 5, no. 3, pp. 301–308, Sep 1997.

[25] K. Kadirvel, J. Carpenter, P. Huynh, J. M. Ross, R. Shoemaker, and B. Lum-Shue-Chan, "A Stackable, 6-Cell, Li-Ion, Battery Management IC for Electric Vehicles With 13, 12-bit (sum) (Delta) ADCs, Cell Balancing, and Direct-Connect Current-Mode Communications," *JSSC*, vol. 49, no. 4, pp. 928–934, Apr 2014.

[26] M. Dave, M. Jain, S. Baghini, and D. Sharma, "A Variation Tolerant Current-Mode Signaling Scheme for On-Chip Interconnects," *TVLSI*, vol. 21, no. 2, pp. 342–353, Feb 2013.

[27] A. D. Mehta, Yao-Ping Chen, N. Menezes, D. F. Wong, and L. T. Pilegg, "Clustering and load balancing for buffered clock tree synthesis," *ICCD*, Oct 1997, pp. 217–223.

[28] G. S. Brodal and A. G. Jørgensen, "A Linear Time Algorithm for the k Maximal Sums Problem," *Mathematical Foundations of Computer Science*, Aug 2007, pp. 578–586.

[29] NCSU, "FreePDK45," http://www.eda.ncsu.edu/wiki/FreePDK45.

[30] G. E. Tellez and M. Sarrafzadeh, "Minimal buffer insertion in clock trees with skew and slew rate constraints," *TCAD*, vol. 16, no. 4, pp. 333–342, Apr 1997.

[31] K. Absel, L. Manuel, and R. K. Kavitha, "Low-Power Dual Dynamic Node Pulsed Hybrid Flip-Flop Featuring Efficient Embedded Logic," *TVLSI*, vol. 21, no. 9, pp. 1693–1704, Sep 2013.

Riadul Islam received his B.Sc. degree in electrical and electronic engineering from Bangladesh University of Engineering and Technology, Dhaka, Bangladesh, in 2007, and the M. A. Sc. degree in electrical and computer engineering from Concordia University, Montreal, Canada, in 2011. From 2007 to 2009, he worked as a full time faculty in the department of electrical and electronic engineering of The University of Asia Pacific, Dhaka, Bangladesh. Currently he is working towards his Ph. D at the University of California Santa Cruz in the Computer Engineering department. His research interest includes low-power clock network design, variability-aware low-power/high-speed digital/mixed-signal circuit design and fault tolerant memory/flip-flop design.

Matthew R. Guthaus is currently an Associate Professor at the University of California Santa Cruz in the Computer Engineering department. Matthew received his BSE in Computer Engineering in 1998, MSE in 2000, and PhD in 2006 in Electrical Engineering all from The University of Michigan. Matthew is a Senior Member of ACM and IEEE and a member of IFIP Working Group 10.5. His research interests are in low-power computing including applications in mobile health systems. This includes new circuits, architectures, and sensors along with their application to mobile and clinical health systems. Matthew is the recipient of a 2011 NSF CAREER award and a 2010 ACM SIGDA Distinguished Service Award.