Received December 31, 2020, accepted January 15, 2021, date of publication January 20, 2021, date of current version January 27, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3053052

Practical Full Chip Clock Distribution Design With a Flexible Topology and Hybrid Metaheuristic Technique

ENG KEONG TEH[1,2], MOHAMAD ADZHAR MD ZAWAWI[1], MOHAMED FAUZI PACKEER MOHAMED[1], AND NOR ASHIDI MAT ISA[1]

[1]School of Electrical and Electronic Engineering, Engineering Campus, Universiti Sains Malaysia, Nibong Tebal 14300, Malaysia
[2]Intel Microelectronic Malaysia Sdn. Bhd., Bayan Lepas 11900, Malaysia

Corresponding author: Nor Ashidi Mat Isa (ashidi@usm.my)

This work was supported by the Ministry of Higher Education, Malaysia, through the Fundamental Research Grant Scheme, under Grant 203/PELECT/6071398.

ABSTRACT This article recommends a practical technique to design full chip (FC) clock tree of a complex system-on-chip (SoC). In the new business environment, the market prefers a highly integrated but low power SoC with fast design productivity and low development cost. We observed that many techniques proposed in the prior arts are no longer practical or enough. With that, we introduce a flexible FC Clock network topology and a synthesis algorithm that utilize a hybrid meta-heuristic technique to search for near optimum solution in shorter turn-around time. Our work on a 10nm SoC product showed that the topology and algorithm managed to produce averagely 16.98% better global skew, 42.75% less divergence on critical clock paths and with 64.5% shorter clock balancing phase compared to a conventional ASIC methodology.

INDEX TERMS System-on-a-chip (SoC), clock distribution, deep sub-micrometer, artificial intelligence, hybrid meta-heuristic.

I. INTRODUCTION

Clock distribution uncertainty, which includes skew and jitters, has always been one of the dominant performance-limiting factors to a chip and thus many techniques have been introduced to reduce clock uncertainty. These techniques can be categorized into 2 solution groups, i.e. *topology* and *algorithm*.

Before the advent of sub-micron process technology and SoC, chips are relatively small, researchers have focused more on local skew optimization algorithm. For examples, the research work in [1] has introduced Deferred-Merge Embedding (DME) algorithm to automate the skew minimization process. Other researcher [2] recommended buffer insertion and adjustment of wires width to further reduce the skew. To further improve clock frequency, intended non-zero localized clock skew could be applied on critical timing paths to increase clock period as recommended by few others [3], [4].

The associate editor coordinating the review of this manuscript and approving it for publication was Syed Islam.

However, as the chip size grow bigger with SoC technology, the effort to fine tune the localized clock trees is becoming increasingly significant. Thus, researches have investigated the method of reducing clock skew topologically. Many researchers have recommended a symmetrical clock distribution topology such as full H-tree, grid or meshes as shown in Fig. **1** to distribute clock signal across the whole chip. These structurally symmetrical clock distribution networks are easily implemented and able to produce very low skew as elaborated by previous group of researches [5]–[8]. However, these networks dissipate higher power [9]. To reduce power dissipation of grid and meshes network, researcher [11] has recommended a reduced voltage swing methodology and researcher [12] has recommended the use of clock frequency multiplier circuitry. Both methods caused an increase in the complexity of analysis and consequently adding risk to the design yield. Furthermore, the grid and meshes networks, which are shorted together at the outputs, are harder to be back annotated into static timing analysis as elaborated by researchers [10], [35].

With the advent of sub-micron technology, the On-chip-variations (OCV), which includes process, voltage,

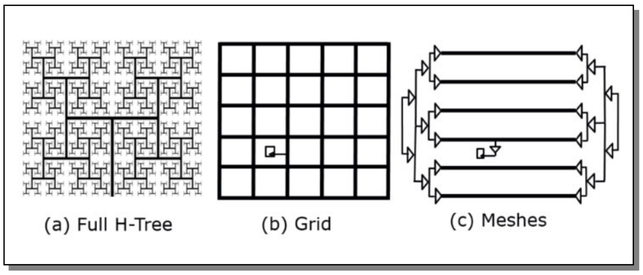

FIGURE 1. Examples of structurally symmetrical clock distribution networks [5], [9], [10], [35].

and temperature (PVT) variations, have big impact on the clock jitter [13], [14]. The impact is even more significant for chip that operates at high clock frequency. As a result, researchers have investigated jitters minimization on top of reducing clock skew. To minimize jitter topologically, structurally symmetrical design [13], [15] and dynamic de-skew circuit [16]–[18] have been recommended to compensate the effect of PVT variations. An example of the clock de-skew distribution networks is shown in Fig. **2**. Even though dynamic de-skew circuit can produce both low skew and OCV induced jitters, to design and validate a dynamic circuit, it requires high skillset and efforts that are not trivial. Similar limitations happen to other complex topology solutions such as differential signal circuitry [19], global resonant H-tree [20], current-mode [68], and hybrid radio frequency (RF)/Metal clock routing [21], [22].

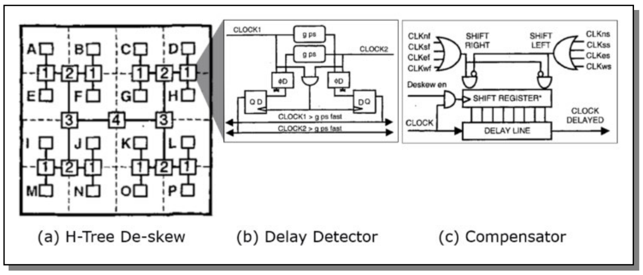

FIGURE 2. A dynamic clock de-skew distribution network [17].

In the new business environment, the market prefers a highly integrated but low power SoC with fast design productivity and low development cost as listed in challenges and possible solution Section of ITRS 2.0 2015 [23]. A highly integrated SoC usually consists of many layouts of hardened intellectual properties (IP) and synthesizable soft IP. The hardened IPs are usually obstacles to the clock distribution network. Besides, different IPs may operate under different clock frequencies and power domains. Thus, the complete topology-based solutions such as grid or meshes networks are no longer a preferred solution for a complex SoC, as on top of higher power dissipation, they consume higher routing resources. As a result, many researchers [7]–[9], [11], [33], [38]–[40], [67], have invested into multi-hierarchy hybrid solutions, which combine topological solutions with clock tree synthesis (CTS) algorithm. An example of hybrid solutions is shown in Fig. **3**. CTS is an automation of Steiner tree construction. Compared to grid and meshes, CTS requires lesser routing resources and has faster turn-around-time as well as lower power dissipation. However, it produces higher global skew [9], [35]. Therefore, CTS is preferred for block level distribution while custom built topological symmetrical trees are preferred at top level [39].

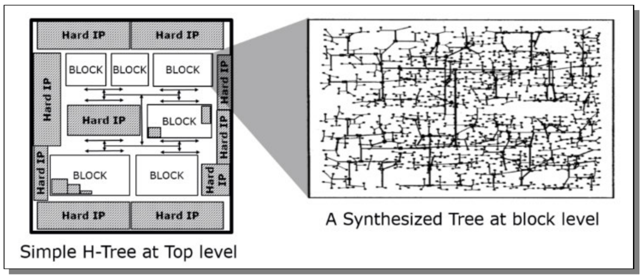

FIGURE 3. A hybrid solution which consist of Simple H-tree at top level and synthesized tree [24] at block level.

As mentioned, hardened IPs and voltage islands can be the obstacles to clock network and thus custom building of a topologically symmetrical tree has become increasingly difficult for a complex SoC. Therefore, fully automated methods are needed to address the complex top-level clock distribution problem [39]. To address this challenge, obstacle aware algorithms CTS [25]–[28] have been recommended. However, OCV induced jitters issue, which is one of the key challenges in deep sub-micron technology with respect to chip design as described in article [29], is not covered in the four above mentioned works. Even though there are prior works that tried to model the OCV hotspots and then pre-optimize the clock design accordingly, many are not practical. For examples, there is a research [30] which recommended algorithms that re-structure an existing clock tree topology to compensate for temperature effects. However, it is hard to obtain an accurate temperature profiling without a fabricated chip. Moreover, temperature changes whenever activity changes and thus it is extremely difficult to alter the design structurally to cover every possible scenario. Similar case happens in the work proposed by researcher [31], in which clock trees are synthesized based on supply voltage drop profile. Clock buffer is one of the main factors of voltage drop. It is a "chicken and egg" situation as accurate voltage drop profile will not be available without clock trees and voltage drop profile will change whenever clock tree changes. Researchers [68] have recommended global current-mode and local voltage-mode clock distribution networks to reduce power consumption and jitter effects. However, it is not practical to implement and validate current-mode circuitries for a complex SoC which has hundreds of clock domains with each have from thousands to millions of sink points. Thus, a more commonly practiced algorithm used to reduce OCV effects on SoC nowadays is by maximizing the symmetrical of the clock topology and minimizing the divergence of the clock trees [32], [33]. Nevertheless, the two above mentioned works cover only a few minor optimizations on divergence reduction algorithm but not solution for the other complex challenges.

Clock design challenges for a complex SoC nowadays are very different from center processing unit (CPU) and small application specific integrated circuit (ASIC). On top of the common quality challenges such as skew, jitter, slew rate, power dissipation etc., there are other important challenges such as newly emerged design complications of a very large complex SoC as well as the tight schedule and resources requirements. Of all the above-mentioned techniques, none of them proposes a complete solution that practically address both top and local levels clock design as well as FC clock balancing challenges. In this article, we recommend a proven practical FC clock distribution solution on a complex SoC. Our key contributions include:

1. A flexible FC clock distribution topology that not only able to handle the recent SoC design complexities but enable parallel executions to align with the tight SoC development schedule.
2. A hybrid-metaheuristic based global clock synthesis algorithm that performs multiple objectives optimization, including obstacle avoidance and OCV reduction, at the same time.

The rest of this article is organized as follows: In Section **II**, we explain the recent FC clock design challenges for a complex SoC in the current business environment. In Section **III** and **IV**, we elaborate the key contributions abovementioned and in Section **V**, we will prove the methodology with the real data extracted from a taped-out SoC product. Finally, Section **VI** concludes the paper.

II. RECENT FC CLOCK DESIGN CHALLENGES

The commonly known design quality challenges of a clock distribution are clock uncertainty which is an aggregation of skew and OCV induced jitter, transition/slew, latency, duty cycle, area, routing resources, signal integrity/crosstalk, reliability, and power dissipation [34]. Among them, clock uncertainty, which is the key performance limiting factor, is the most focused area. Clock power consumption is also one of the key challenges as it is often the largest portion of total chip power. For example, the clock driver and pre-driver represent about 40% of the total effective switching capacitance determined by power measurement in the Alpha processor which runs at 200MHz [45].

Researchers [35] evaluated and compared quantitatively different clock architectures such as mesh, tree and their hybrids, on three industrial designs with respect to latency, skew, timing uncertainty and power. As concluded by [35], mesh architecture was more effective than a tree architecture in obtaining very small maximum clock skew (below 1ps) on all the designs. A finer mesh is more effective than a coarse mesh. However, there can be an up to 30% power penalty due to mesh. Power can be saved by employing the Tree + Local meshes (TLM) architecture with multiple meshes, each of which could be independently disabled. A mesh-based architecture is more robust to variations and can reduce the timing uncertainty due to parameter variations in the global tree by 18%.

Other than that, researcher [44] has also compared different clock distribution techniques used in different processors as shown in Table **1**. Whereas Table **2** shows the skew, in which it accounts on average for about 5% of the cycle time and is trending higher as frequency increases [44].

TABLE 1. Comparison of Different Clock Distribution Techniques [44].

Technique	*Wire Cap*	*Delay*	*Skew*	*Processors*
Grid	*High-15x*	*Low-sub100ps*	*Low-Med*	*SPARC, Alpha*
Symmetrical Trees	*Low-1x*	*High-100's ps*	*Low*	*IBM & Motorola Power-PC, HP PA-RISC*
Serpentine	*Very High-30x*	*High-100's ps*	*Low*	*Pentium-III*
Spine	*High-10x*	*Low-sub100ps*	*Med*	*Alpha, Pentium-4*

TABLE 2. Industry Clock Skew Data [44].

Processor	*Frequency (MHz)*	*Clock Skew(ps)*	*Percentage of Cycle Time (%)*	*Process*
Itanium™	*800*	*110ps w/o de-skew* *28ps w/ de-skew*	*8.80* *2.24*	*.18um*
PowerPC	*1000*	*15ps with Cu wires*	*1.50*	*.22um*
UltraSPARC III	*800*	*80ps Al wires, no de-skew*	*6.40*	*.18um*
Alpha	*600*	*72ps Al wires, no de-skew*	*4.32*	*.13um*

Other than the design quality challenges above mentioned, in the recent business environment, the following challenges have also emerged as the primary concerns:

- Design Complexity - As mentioned by researchers [36], in the 2013 ITRS roadmap, the die area of Consumer Portable SoC (SOC-CP) is about 140mm^2 and transistor count is about 2.4 billion transistors. Recent trends suggest a factor of 1.26x scaling of logic transistor per core with every process technology node.
- Design productivity and development cost – As listed in challenges and possible solution Section of ITRS 2.0 2015 23], the key system integration long term (> 3 years) challenge is design productivity where it is beneficial for faster design turn-around-time and less design effort.

In recent years, the numbers of IP-cores integrated in a SoC has continuously increased. In these complex SoCs, multiple clock domains are required in order to feed difference frequency clock signals to individual IP-cores [41]. On top of engineering efforts, the routing resources that are needed to distribute and shield the clocks have increased. Early routing resource planning is required to prevent late discovery of congestion issue which will impact the project schedule consequently. Besides, in a large SoC, these clock signals need more repeater stages to travel long distances and their distribution patterns can be non-systematic as illustrated in Fig. **4**. As the chip size continues to grow, more

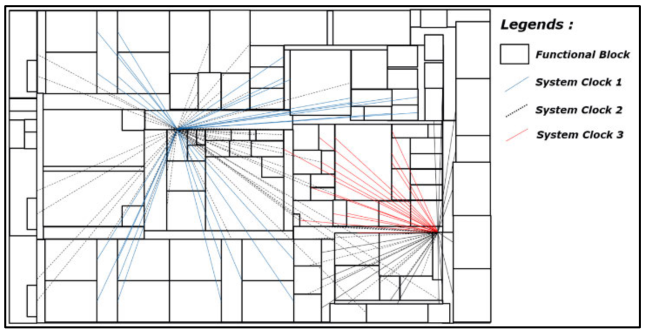

FIGURE 4. An example of multiple clock signals distributed in a non-systematic pattern at top level of a complex SoC.

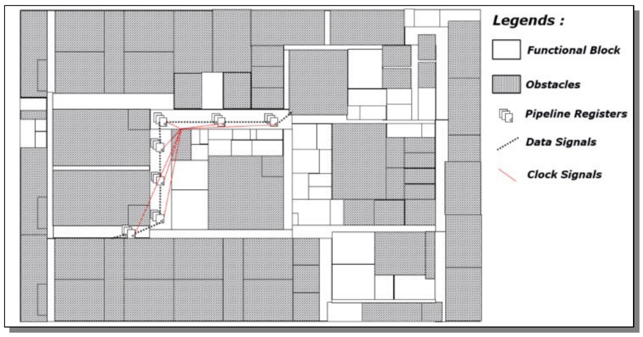

FIGURE 6. An example of pipeline registers insertion at top level avoiding the obstacle.

clock repeater stages will be needed and consequently OCV induced period jitter will increase [14].

Besides, the IP-cores could be multiple instantiated blocks (MIBs) at top level. Adding any global clock structure into a MIB instance will cause all the other instances to duplicate the same structure. Thus, MIBs are preferred to be treated as obstacles to simplify FC integration in the later stage. On top of that, some synthesized IP blocks could be either power gated and/or having different power domains (multi-VDD). They are usually being treated as obstacles to avoid insertion of level shifter or self-isolated repeater on the clock networks, which will add complexity into the implementation, balancing, and analysis works. With both MIB and multi-VDD in a complex SoC, obstacles can be in a non-systematic formation as illustrated in Fig. **5**. As a result, different clock domains may have different distribution patterns and routing paths. Therefore, symmetrical networks such as grid and meshes are no longer practical for top level clock distribution in a complex SoC nowadays.

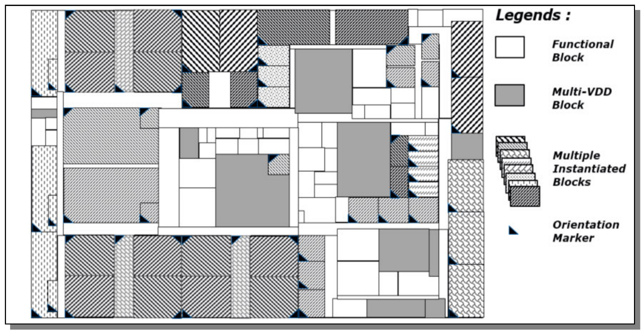

FIGURE 5. An example of FC floorplan of a SoC with flipped MIBs and multi-VDD blocks.

As complex SoCs grow larger in size and higher in clock frequency, the flop-repeaters, which are inserted to split a timing path to gain additional timing margin, has become almost a necessity [42]. These flop-repeaters will be inserted into channel or non-obstacle blocks following the signal routing path as illustrated in Fig. **6**. In a recent complex SoC, the amount of timing paths that require flop-repeaters can be large and up to hundred thousand of flop-repeaters may be needed [43]. Furthermore, these paths are driven by different clock domains from various source IPs to destination IPs. In other words, the signal routes and flop-repeaters placement are non-systematic. Therefore, clock distribution network of these flop-repeaters has emerged as a new challenge to a SoC design.

Other than design complexity, design productivity and development cost are the recent challenges in a SoC. As proposed by researcher [37], to ensure a predictable productivity, a workflow that is cognizant of the maturing nature of a design over time is needed. Logic and physical design are executed in parallel. Milestones are defined by the maturity of the netlist, IP, and layout. As a result, design cycle is commonly divided into phases according to the design maturity model. For example, design cycle has been divided into four phases, i.e. setup, convergence, refinement as well as closing [37]. Every phase has different input requirements namely start-of-phase deliverables, and end goals as shown in Table 3. Clock specification and design maturity are important input requirements to the phases. Thus, FC clock design workflow must be planned properly to avoid gating the progress of both top and block level physical implementations as well as the FC timing convergence. Beside a realistic design productivity plan, putting together the right resources (team skill composition, machines and licenses) and managing them efficiently is also one of the four key issues that dictates the success of a SoC project [37]. In other words, we need to manage tradeoffs properly in order to come out with a FC clock design methodology which is at the right complexity level and achievable with the engineering resources allocated.

In summary, with the newly emerges design complexities, fast design productivity and low development cost requirements, many prior FC clock design techniques that focus on quality of result (QoR) improvement only are no longer enough. In this article, we focus on development of a practical FC Clock design methodology, which include both topology and algorithm solutions, to solve the challenges above mentioned.

III. A FLEXIBLE FC CLOCK TOPOLOGY

In this section, we introduce a flexible FC clock distribution topology, which is a hybrid trees that used to solve the

TABLE 3. Deliverables and Goals for Each Phase of The Design Maturity Workflow [37] (A Reduced Version).

Items	*Setup*	*Convergence*	*Refinement*	*Closing*
Description	*Setup the rest of the project Coordinate the timing of specific events. Assemble Resources*	*Stabilize design to build the chip. Build FC from deliverables. Expect multiple netlist & IP iterations*	*Realize the final SoC specification.*	*Close the design. Complete the project to tape out to the mask ship*
Start-of-Phase Deliverables				
Netlist	*Initial*	*Structurally Completed. All IPs in. Top level Stable*	*Almost Final. Final very minor tweaks.*	*Final*
IP, cell lib. & mem.	*Initial*	*Updated*	*Final Version*	
Spec.	*SoC Information*	*Updated (I/O, Clocks, Timing)*	*Final I/O*	
Constraints	*Project (Die size, Power)*		*Final Timing*	
End-of-Phase Goals	*Project plan & Project setup completed. IP setup and checked. Netlist analyzed. Trial floorplan. Trial block build.*	*Top Level Stable (Pads, Power Grid, Floorplan). All blocks implemented. Initial timing run, timing constraints, power rail analysis. Formal verification passed. Block level DRC/LVS runs. Die size convergence.*	*All blocks are routable. FC timing closable. Most DRC/LVS cleanable. All RV issues resolved. Top-level frozen. Initial FC layout checks.*	*Final GDSII build. Minor timing ECOs. Final Analysis. Final tape out database created. Final verification and fixes.*

recent SoC complexities as elaborated in Section **II**, i.e. high frequency clocks design requirements, multiple widely distributed clock domains, many flipped MIBs, non-systematic flop-repeaters insertion, as well as fast time-to-market and low development cost requirements. The architecture of the proposed FC clock topology, which is a hybrid solution, is shown in Fig. **7a**. The proposed topology consists of 3 key components, i.e. global clock trunk, local CTS, and FC Clock Balancing Circuitry as in Fig. **7b**. Basically, global clock trunk is used to distribute clock signals evenly from Clock Source (PLL) to regional drop-off-points (DOPs) while local clock CTS is used within block level to distribute clocks to the sequential logics such as latches, flops, and memory macros. In between them, FC Clock balancing circuitry is used to balance the total clock latency between blocks.

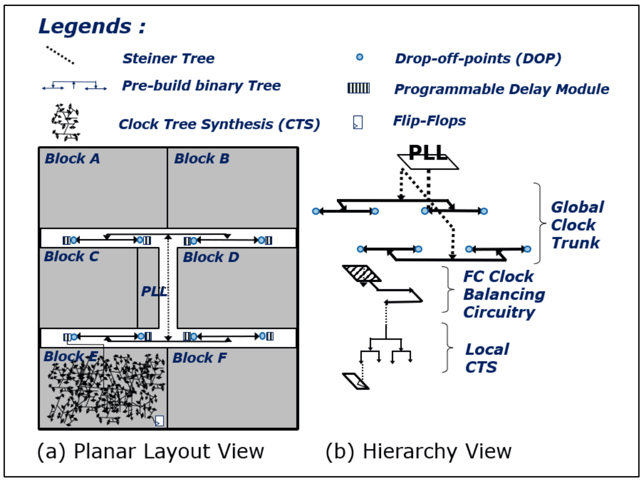

FIGURE 7. The architecture overview of the proposed multi-hierarchy hybrid FC clock topology.

Explanation on how these components overcome the new design complexities are in the next sub-sections.

High level of the FC clock design flow is shown in Fig. **8**. It is designed to align smoothly with the design phases as illustrated in Table **3** and consequently ensure fast productivity and minimize reworks.

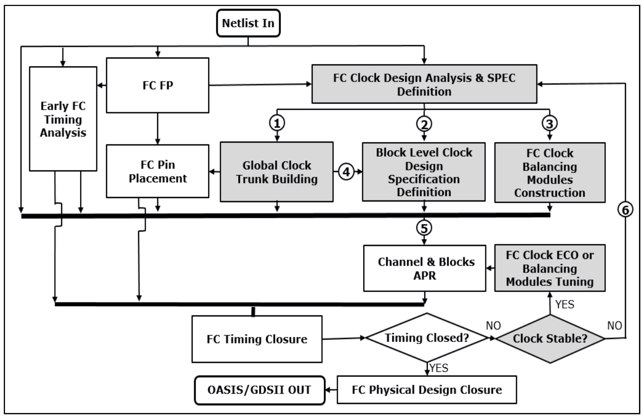
Netlist In
Early FC Timing Analysis
FC FP
FC Clock Design Analysis & SPEC Definition
1
2
3
FC Pin Placement
Global Clock Trunk Building
4
Block Level Clock Design Specification Definition
FC Clock Balancing Modules Construction
6
5
Channel & Blocks APR
FC Clock ECO or Balancing Modules Tuning
YES
FC Timing Closure
Timing Closed?
NO
Clock Stable?
NO
YES
OASIS/GDSII OUT
FC Physical Design Closure

FIGURE 8. The proposed parallel FC clock design flow.

During "Setup" phase, based on the trial FC floorplan (FP), the logical connectivity extracted from the initial netlist, system level specifications, and the maximum clock frequencies, we will propose the initial top-down clock implementation specifications, such as latency, skew and slope targets for the three key components. The targets will be fine-tuned based bottom up feedback after trial designs of the components. To reduce the design turn-around-time (TAT), the 3 components are designed in parallel. During "Convergence" phase, multiple iterations of netlist and IP releases will happen, especially block level netlist. Thus, CTS, which trades off QoR for faster TAT, is recommended for block level. The CTS engine is configured based on specification defined through prior evaluation works which will be explained in Sub-section **B**. Global clock connectivity at top level, which is basically the connections between PLL and blocks, will be relatively more stable compare to the local clock connections at block level. Thus, we propose to build the global clock trunk to achieve higher QoR as elaborated in Sub-section **A**. Meanwhile, clock balancing module, which is programmable with simple engineering change order (ECO) effort, will be constructed in preparing for fast tuning in the following phase. Detailed explanation of the modules is in Sub-section **C**. During "Refinement" phase when both global and block level clocks are stable, we will focus mainly on programming the balancing modules to ensure fast TAT and minimum impact to the overall system.

A. HYBRID GLOBAL CLOCK TRUNK

The primary objective of the proposed global trunk is to distribute clock from PLL to the regional DOPs with minimum clock latency and skew. Besides, slope, number of diverged stages that impact the OCV, maximum capacitance that impact the reliability, and duty cycle distortion need to be considered during construction to reduce design reworks which slow down the design productivity. A complex SoC may have non-systematic obstacles as shown in Fig. **5**, in which grid and meshes based networks are not easily implemented. CTS is more flexible in handling design with non-systematic obstacles but with trade off on skew and jitters. To solve the dilemma, we proposed to use a hybrid global clock structure in which it combines DOP trees that are built based on symmetrical binary/H-trees with a synthesized Steiner tree as shown in Fig. **7**.

As mentioned in Section **II**, one of the recent challenges in a complex SoC is the clocking of flop-repeaters. A timing path with flop-repeaters is naturally a source synchronous path where a dedicated clock signal is traveling side-by-side with the data bus from source to the destination as illustrated in Fig. **9a**. However, in a complex SoC, hundred thousand of timing paths may need flop-repeaters and these flops are inserted into non-obstacle blocks or channels in non-systematic patterns. If every flop-repeated path has its own dedicated clock signal traveling side-by-side, there will be massive amount of clock nets introduced. This will not only increase the complexity of clock modeling in static timing analysis (STA) but also the routing resources and power dissipation significantly. On top of that, the timing path driven by the last stage of flop-repeater will become critical due to large clock skew as shown in Fig. **9a.** To simplify the clocking of these non-systematic flop-repeaters and avoid the critical timing path, we propose to use a Balance Clocking Scheme (BCS) as shown in Fig. **9b** or a Useful-Skew Clocking Scheme (UCS) as in Fig. **9c**. BCS has the benefits of simplicity but longer total clock latency compare to UCS. This is because, both blocks and flop-repeaters in BCS are driven by DOPs but clock latency within blocks are usually larger. To balance the latency between the twos, we will need to add clock delay into path of the flop-repeaters. The added delay can be large in order to balance with a larger block. In contrast, with UCS, blocks will not be driven by DOPs but the earlier clock brunches. Useful skew can be applied between DOPs and input clock pin of the blocks to compensate the latency gap between flop-repeaters and the blocks. As a result, latency from DOPs to flop-repeaters can be reduced and consequently reduce the total diverged latency which directly contributes to the OCV induced jitter [14]. In conclusion, it is recommended to use the simpler BCS by default for faster design productivity but UCS for the large blocks.

The proposed global clock trunk design flow is shown in Fig. **10**. First, an optimum clock repeater recipe will be identified using metaheuristic technique as proposed in our previous work [43], where a Genetic Algorithm (GA) based flow will perform global search through auto mixing and matching the input variables (aka "Genes") such as repeater types, wire layer/width/spacing, and repeater interval distances to perform layout constructions (aka "reproduction"),

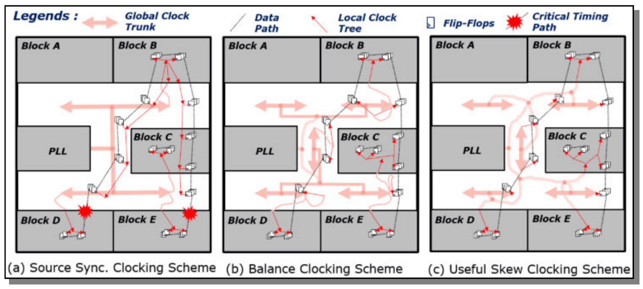

FIGURE 9. Comparison of proposed clocking schemes for flop repeaters with the source sync. scheme.

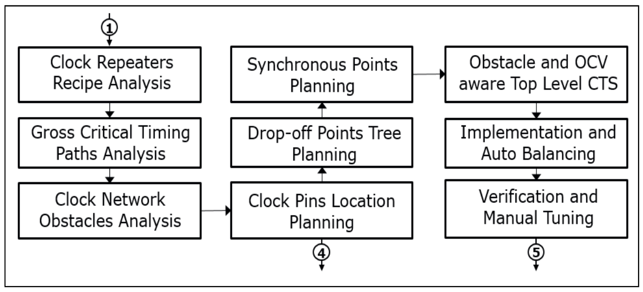
1
Clock Repeaters Recipe Analysis
Gross Critical Timing Paths Analysis
Clock Network Obstacles Analysis
Synchronous Points Planning
Drop-off Points Tree Planning
Clock Pins Location Planning
4
Obstacle and OCV aware Top Level CTS
Implementation and Auto Balancing
Verification and Manual Tuning
5

FIGURE 10. The proposed global clock trunk design flow. (Note: Refer to Fig. 8 for definition of number 1, 4, and 5).

and performance analysis (aka "Selection"). Based on the data paths information provided by logic designers, partitions physical location provided by FC floorplan, and the repeater latency recommended by the GA based recipe finder flow, critical FC timing paths will be identified, and higher weightage will be applied on the cost of clock divergence point of these paths during top level Steiner tree synthesis later. With the information of MIB and power well information, non-obstacle blocks, which clock feedthrough is allowed, will be identified. The non-obstacle blocks together should form continuous routes which allow clock to distribute from sources to every functional block. At the same time, clock pins of functional blocks should always be planned to face the non-obstacle blocks. Based on the clustering of the clock pins, DOP tree location will be planned. At this stage, clock latency data should have been extracted from the earlier block which run with an automated place and route (APR) flow. Based on the data, synchronous points which offset the block level latency will be modeled. With the critical paths, obstacle, and synchronous point information, we will perform a top level OCV aware Steiner tree synthesis using a k-mean meta-heuristic technique [55] which will be elaborated in Section **IV**. After implementation and balancing of the trunk to below 50ps skew target, the trunk and the pre-built programmable delay modules, which will be discussed in Sub-section **C**, will be pushed down into block level.

B. MINIMUM LATENCY TARGETED LOCAL CTS

Larger clock latency from a divergence point will cause larger OCV induced clock jitter. Thus, to reduce jitter, it is recommended to minimize the clock latency at block level. To achieve that, an evaluation phase in the design cycle has been initiated to collect the minimum latency achievable by each block. The flow used in the proposed block level CTS evaluation phase abovementioned is shown in Fig. **11**. During the evaluation, CTS will begin with default top-down clock design specification, pin locations, and configuration which prioritizes on minimizing of clock latency. Then, multiple iterations of CTS trial run, QoR analysis, and tool configuration tuning will be performed until acceptable skew has been achieved. The clock latency data collected from each block will be set as the clock design target for its future APR. At the same time, balancing modules will be pre-built/pre-inserted to offset gross latency gaps between blocks.

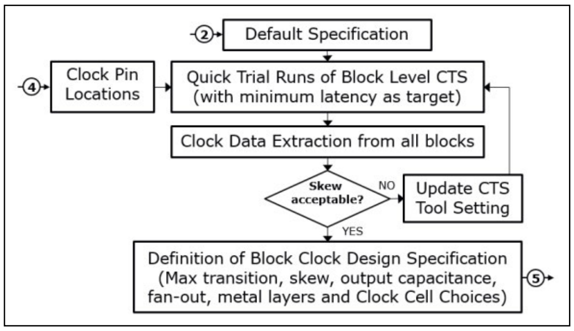
2
Default Specification
4
Clock Pin Locations
Quick Trial Runs of Block Level CTS (with minimum latency as target)
Clock Data Extraction from all blocks
Skew acceptable?
NO
Update CTS Tool Setting
YES
Definition of Block Clock Design Specification (Max transition, skew, output capacitance, fan-out, metal layers and Clock Cell Choices)
5

FIGURE 11. The proposed minimum latency targeted block level local CTS flow. (Note: Refer to Fig. 8 for definition of number 2, 4, and 5).

One of the new clock design complexities is related to the challenges of clock balancing for blocks that have the concurrent clock data (CCD) optimization feature turned on. CCD feature uses "useful" skew, which is a clock skew intentionally added to meet timing, to optimize critical timing path as shown in Fig. **12**. Additional delay buffers are added into clock paths of receiving flops to skew clock intentionally in order to gain more timing margin and consequently resolve setup violations. Noted that we must specially configure the APR tool and flow to avoid adding delay into clock path of the input/output (IO) flops, as it will complicate the cross blocks clock balancing and FC timing convergence tasks. Besides, the APR tool may not able to identify IO flops accurately using its built-in command, as it could be confused by design for test (DFT) signals that connect ports to every flop. To mitigate the issue, we must filter out the IO flops from other flops with automation scripts based on special naming convention.

To further minimize the clock latency especially in non-obstacle blocks where flop-repeaters could be widely and randomly distributed, multi-source CTS (msCTS) with low latency pre-built trunk/mesh is used as shown in Fig. **13**. msCTS is a hybrid method of conventional CTS and Clock mesh. Multisource drivers connect to the trunk/mesh at a limited number of locations referred to as tap points. A multi-source clock tree structure driven by the mesh consists of sub-trees, each driven by a tap point [56]. By using low skew and latency pre-built trunk/mesh, we can reduce the total clock skew, latency and OCV effects.

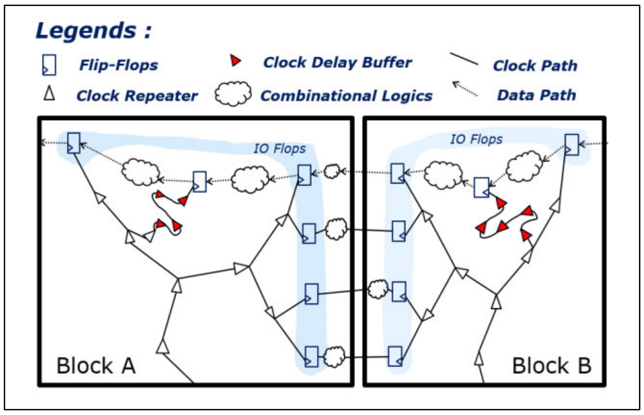

FIGURE 12. Designs with CCD feature, in which useful skew is automatically applied by CTS engine to improve timing.

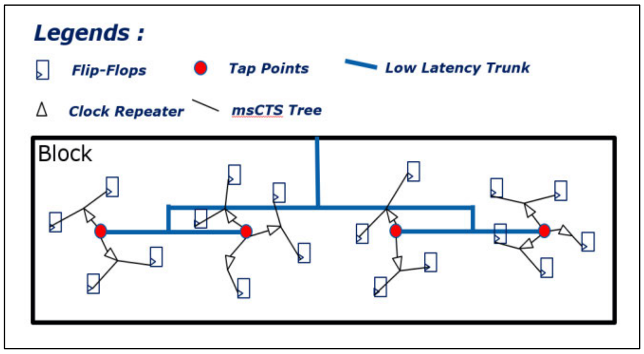

FIGURE 13. Structure of proposed multisource CTS network which is meant to reduce clock latency.

C. FC CLOCK BALANCING CIRCUITRY

FC clock balancing usually happen after the global and block level local CTS clocks are stable. However, it is proposed to preemptively reduce the gross skew since the "Setup" phase (see Table 3) during trial block build process. Clock delay modules and insertion flow are pre-built based on the data collected from block level CTS evaluation phase as mentioned in Sub-section **IIIB**. There are 2 types of balancing modules proposed in this article: (1) Zig-zag buffer Chain (ZZB) and (2) Programmable Delay Module (PDM). ZZB is used for preemptive balancing while PDM is used for final clock latency fine tuning.

ZZB, as shown in Fig. **14**, is basically a simple clock repeater chain which is designed to mimic the structure of CTS clock tree in block level, where the most commonly used clock repeater types, wire layers, and interval distance between clock repeaters are chosen. With the similar clock structure, it can help to preserve similar net delay to cell delay ratio across clock branches and consequently reduce the global clock skew as process, voltage, and temperature (PVT) changes. ZZB is used to offset gross latency gap of smaller blocks. Each ZZB will provide approximately 450ps delay per instantiation. An automation script has been developed to parse the data collected from the block CTS evaluation

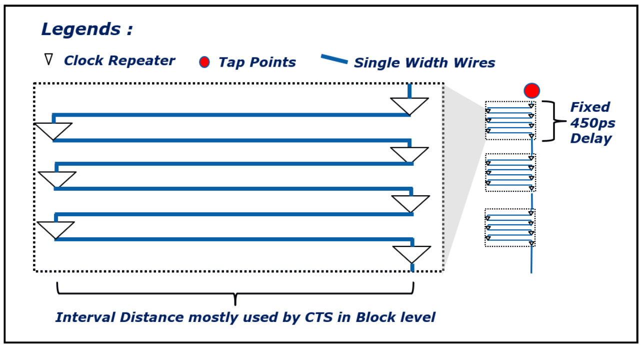

FIGURE 14. Proposed Zig-Zag Buffer (ZZB) chain that is having similar net delay to cell delay ratio as in block level CTS.

phase as described in Sub-section **IIIB** and auto generate ZZB insertion Engineering Change Order (ECO) scripts for non-obstacle blocks or channel partitions to source in during APR of block.

The proposed PDM, as shown in Fig. **15**, is basically a simple chain of multiplexers which is carefully designed to produce balance duty cycle.

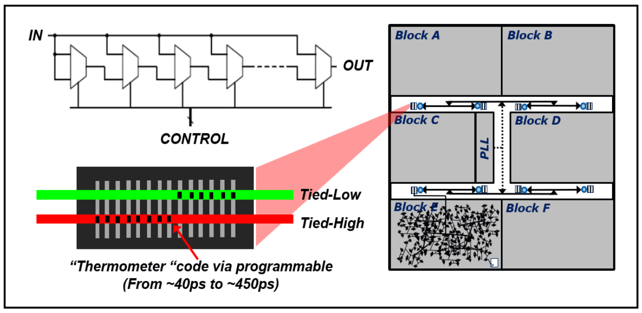

FIGURE 15. Basic design concept of the proposed programmable delay module (PDM).

PDM controlled with thermometer code (TC) format as shown in Table **4**, where the value of each number is expressed as the number of ones in a string of bits [62]. PDMs are meant for clock latency fine tuning after stabilization of both global and local clock tree. In our work, they can be programed to provide approximately 40ps to 450ps delay with 40ps of granularity at typical PVT corner.

TABLE 4. Delay versus Thermometer Code (TC).

Regular Representation	*Control Bits*	*Approximate Output Delay Range(ps)*
0	*00 0000 0000*	*40-50*
1	*00 0000 0001*	*80-90*
2	*00 0000 0011*	*120-130*
3	*00 0000 0111*	*160-170*
4	*00 0000 1111*	*200-210*
5	*00 0001 1111*	*240-250*
6	*00 0011 1111*	*280-290*
7	*00 0111 1111*	*320-330*
8	*00 1111 1111*	*360-270*
9	*01 1111 1111*	*400-410*
10	*11 1111 1111*	*440-450*

The programming can be achieved either with via programmable scripts or routing ECO feature of the APR tool to tie the select signals of the multiplexers to the corresponding logic ones or zeros.

Some of the non-obstacle blocks or channel partitions may contain tap points or DOPs, where global trunks have distributed clock signals from PLL across chip to the regional stations for functional blocks to connect to for clock source. PDMs are proposed to be inserted for each clock domain of each block at tap points, which are basically the leaf nodes of DOP trees, as shown in Fig. **16**. This will increase OCV jitter on timing paths across blocks but to have better control during final clock balancing phase, which is critical to the overall design converging schedule, the tradeoff is highly recommended. Every clock domains of every blocks will have their own PDMs but not ZZB. ZZB is inserted on need basis. Thus, ZZBs are recommended to be placed after PDMs to preserve consistency of capacitance load driven by tap points.

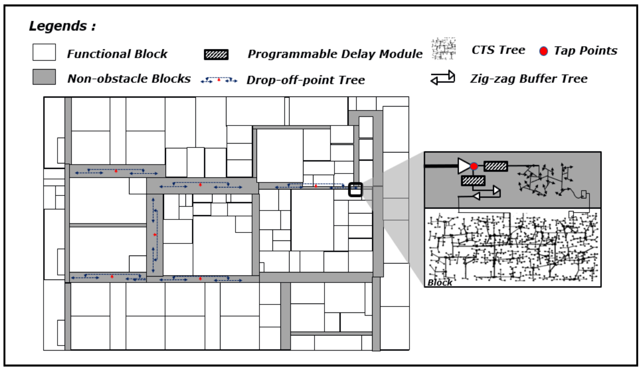

FIGURE 16. Topological view of the proposed clock balancing circuitry.

IV. HYBRID METAHEURISTIC BASED GLOBAL CLOCK SYNTHESIS ALGORITHM

As elaborated in Section **I**, most of the prior global clock synthesis techniques did not consider performance-power-area (PPA) and OCV at the same time. To achieve global optima solution, global clock synthesis of a complex SoC can be a non-deterministic polynomial-time hardness (NP-Hardness) optimization problem, where no algorithm exists to solve it in polynomial time. The usage of exact algorithms, such as mathematical based approaches, to give exact or complete solutions is impractical. In approximation or namely heuristic methods, the guarantee of finding optimal solutions is sacrificed in order to get near-optimum solutions in reasonable and practical computational times [61]. In global clock synthesis context, an example of heuristic method is to configure constraints of a layout computer aid design (CAD) tool and then rely on its CTS algorithm to build the solution. CTS deploys its built-in meta-heuristic algorithm, which is defined as an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space [57], to search for solution which focuses on minimizing skew and latency. However, without awareness of the contents within the block level and the timing relationship between blocks, CTS is just a local search algorithm which has no means by which it can search extensively and exploit global and optimum solutions found in the neighborhood existing in distances of the solution space [59]. Therefore, local search algorithm will simply get caught in the local optima [61]. Some CAD tools have provided feature to include full or partial blocks' content into the CTS. However, not all contents will be available in time and even they are, the run-time of global CTS may not be acceptable, especially for complex SoC that has more than millions of flops. Furthermore, the complex on-chip clock controllers (OCC) built within the blocks will deteriorate the CTS QoR if they are not properly configured. To overcome the challenges, we propose to use the hybrid meta-heuristic algorithm [60] which combines meta-heuristic with some other exact or heuristic or meta-heuristic algorithms to remove each other's weaknesses and merge their strengths.

High level of the proposed overall global clock implementation flow is shown in Fig. **17**. In this flow, we introduce an enhanced hybrid meta-heuristic (HMH) based global CTS algorithm to perform thousands of quick search, which able to analyzed based on incomplete design data set, for a near global optimized design recipe before passing the solution to the standard design flow which requires longer turn-around-time to physically implement the distribution network.

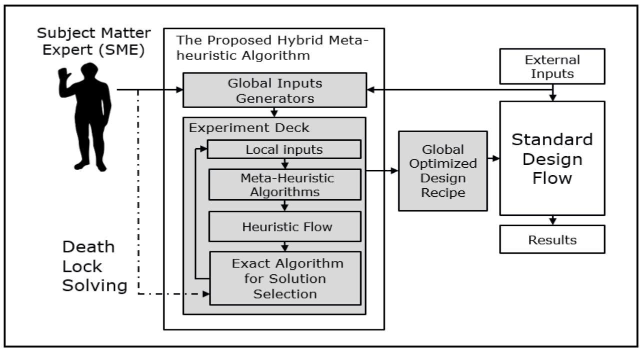
Subject Matter Expert (SME)
The Proposed Hybrid Meta-heuristic Algorithm
Global Inputs Generators
Experiment Deck
Local inputs
Meta-Heuristic Algorithms
Heuristic Flow
Exact Algorithm for Solution Selection
Global Optimized Design Recipe
External Inputs
Standard Design Flow
Results
Death Lock Solving

FIGURE 17. **Overview of the proposed global clock synthesis flow.**

The proposed HMH based global CTS algorithm consists of three key components:

(1) K-mean metaheuristic algorithm, which takes in simplified global inputs that defines the floorplan, pins location, block level estimated clock latency and relationship information to perform global search for the branching points of the distribution network.

(2) Fast heuristic algorithm, which deploys industry CAD tool to perform local search for implementation solution, such as repeater insertion, place, routing as well as shielding, and then estimates the QoR.

(3) Exact algorithm, which models Subject Matter Expert's (SME) solution to control selection process, to avoid death lock, and to solve multi-objectives optimization problem.

Detailed explanation of the three components are in the following sub-sections.

A. K-MEAN BASED GLOBAL CLOCK SYNTHESIS ALGORITHM

K-mean has been used for sink points clustering by researchers [68] and branching points identification by researchers [69]. In this article, the algorithm has been extended to enable global CTS for a hierarchical design that is obstacle dominant at top level. Basically, the proposed global CTS flow constructs tree by identifying branching points and their locations in a stacking format as shown in a simplified model in Fig. **18**. After identifying branching points for the first stage of clock end points, the branching points will become pseudo end points for the following stage. After that, it will become a recursive process until only one pseudo end point left in one stage. Then the end point will be connected to input clock pin. Upon completion of the recursive flow, balance stages of clock repeaters will be inserted as shown in Fig. **18(f)**. After that, the tree is ready to be detail-routed and fine-tuned to improve skew.

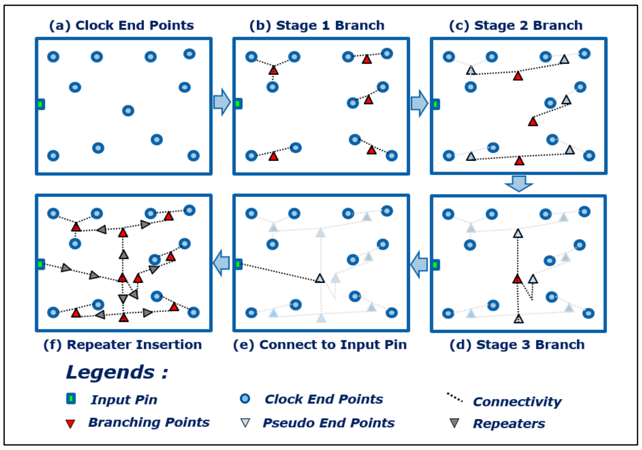

FIGURE 18. **A simple model of the proposed global CTS flow.**

Branching points finding is basically a clustering process. In this work, the k-means algorithm is chosen for this purpose as it is one of the most widely used clustering heuristics [55]. The k-means algorithm solves the problem of clustering to minimize the sum of squared errors (SSE). In global CTS, we are given a set of end points, which are basically the clock end points $P \epsilon \mathbb{R}^d$ in a Euclidean space, and the goal is to find a set of branching points, or centers $C \epsilon \mathbb{R}^d$ of k points (not necessarily included in P) such that the sum of squared distances of the points in P to their nearest center in C in minimized. Thus, the objective function to be minimized is

$$cost\ (P, C) := \sum_{p \epsilon P} \min_{c \epsilon C} \|\text{p-c}\|^2 \tag{1}$$

where $\|.\|^2$ is the squared Euclidean distance. The points in C are called centers or centroids. The above problem formulation assumes that the number of centers k is known in advance [55]. However, in global CTS case, k is not apparent from the application thus we will have to search for the correct number of k. To achieve that, since the number of end points at top level are a lot smaller, we apply brute-force search [63], which systematically enumerating all possible candidates for the solution and a QoR measurement method, which is elaborated in Sub-section **C**, for selection.

In order to solve the SSE problem heuristically, the k-means algorithm starts with an initial candidate solution $\{c_1, \ldots, c_k\} \in \mathbb{R}^d$, which can be chosen arbitrarily (often, it is chosen as a random subset of P). Then, two steps are alternated until convergence: First, for each c_i, the algorithm calculates the set P_i of all points in P that are closest to c_i (where ties are broken arbitrarily). Then, for each $1 < i < k$, it replaces c_i by the mean of $\mathrm{P_i}$ [55]. In the proposed global CTS case, the locations of objects are modeled in Cartesian coordinate system, thus the Euclidean distance calculation will be based on $P_i(x_i, y_i) \in \mathbb{R}^d$ values. The proposed k-mean algorithm is modeled in Algorithm **1** in Fig. **19.**

Legalization of center point is needed to avoid overlap with clock obstacles. It is basically performed by moving any overlapped point to the closest non-obstacle region which identified using Boolean operation on polygons, which is a computational geometry technique [64], as shown in Fig. **20**. To achieve that, all the non-obstacle regions adjacent to the obstacle region which overlapped with the center point will be modeled as combination of rectangular boxes. Lower left (LL) point, (x0, y0) and upper right (UR) point, (x1, y1) of all boxes will be extracted as shown in Fig. **20a**. For each box, by combining with the coordinates of the center point $(\mathrm{x_c}, \mathrm{y_c})$, two new virtual boxes, as labelled A and B in Fig. **20b,** will be generated with coordinates:

$$\begin{aligned} \text{Box A}: \ \text{LL point} &= (\mathrm{x0} - \Delta, \mathrm{y0} - \Delta) \\ \text{UR point} &= (\mathrm{x_c} + \Delta, \mathrm{y_c} + \Delta) \\ \text{Box B}: \ \text{LL point} &= (\mathrm{x_c} - \Delta, \mathrm{y1} - \Delta) \\ \text{UR point} &= (\mathrm{x1} + \Delta, \mathrm{y_c} + \Delta) \end{aligned} \tag{2}$$

where $\Delta = 1\ \mu$m, which is a simple place holder value for overlap region calculation. The overlap region, which is the legal region that is the nearest to the center point, will be calculated using the "AND" Boolean operation on the two virtual boxes. Upon completion of overlap regions identification for all the rectangular boxes of adjacent non-obstacle regions, the region with the shortest distance to the center point will be selected for adjustment as in Fig. **20c**.

By default, for cluster with only one member, k-mean algorithm will always set its center point exactly on top of the member itself. This is referred as back-to-back (B2B) point in this article. This will cause large global clock skew as shown in Fig. **21a**. To overcome this dead-lock issue, first, the overall average Euclidean distance, D_p of all other end points to their respective center points, exclude any cluster with single member, will be calculated. Then, an exact searching algorithm, which is based on a computational geometry technique [64], has been used to find a legal region that is D_p distance from the end points leaning toward center point of

```
Algorithm: 1
Inputs:
  End point set, $P(x,y)= \{p_1(x_1,y_1), p_2(x_2,y_2), \ldots\} \subseteq \mathbb{R}^d$
  Integer number of centers, $k \subseteq \mathbb{Z}$ where $k < |P(x,y)|$
  Initial center point set, $c_i = \{c_1(x_{c1}, y_{c1}), \ldots, c_k(x_{ck}, y_{ck})\} \in \mathbb{R}^d$
Outputs:
  Clusters Sets $P_i \subseteq P(x,y)$ where $i = 1$ to $k$
  Cluster center point Set, $C(x,y) = \{c_1(x_{c1}, y_{c1}), \ldots, c_k(x_{ck}, y_{ck})\} \subseteq \mathbb{R}^d$
  Score, $S \in \mathbb{R}$
repeat
  Empty Set definition for clusters $P_1, \ldots, P_k \leftarrow \emptyset$
  For each end point $p(x_p, y_p) \in P(x,y)$ do
    For each center point $c_i(x_{ci}, y_{ci}) \in \mathbb{R}^d$, where $i = 1, \ldots, k$ do
      Calculate the Euclidean distance, $d_{pi} = [(x_p-x_{ci})^2+(y_p-y_{ci})^2]^{0.5}$
    End for
    Identify the $i$ which has the minimum $d_{pi}$ where $1 < i < k$
    Set $p(x_p, y_p)$ as a member of cluster $P_i$
  End for
  For each cluster $P_i$, where $i = 1$ to $k$ do
    If $P_i \neq \emptyset$ then calculate new center point $C_i(x_{ci}, y_{ci}) \in \mathbb{R}^d$,
      $x_{ci} = \frac{1}{j}\sum_{n=1}^{j} x_n$ where $\{p(x_1, y_1), \ldots, p(x_j, y_j)\} \in P_i$
      $y_{ci} = \frac{1}{j}\sum_{n=1}^{j} y_n$ where $\{p(x_1, y_1), \ldots, p(x_j, y_j)\} \in P_i$
      Legalize the new center point (see Fig. 20)
  Else $P_i = \emptyset$ then
      Abort flow and flag as death lock
  End if
  End for
until the centers do not change

Calculate the overall average Euclidean distance, $D_p$ of all end points
to their respective center points.

For each cluster $P_i$ where $i = 1$ to $k$ do
  if member count of the cluster, $|P_i|=1$ then
    Adjust the back-to-back (B2B) center point to a location which is
the $D_p$ distance away from the end point (see Fig. 21).
  End if
End for
Calculate and Record the Score, S (see Sub-Section C)
```

FIGURE 19. The proposed *k*-mean algorithm to find center points based on predefined initial center points and value of *k*.

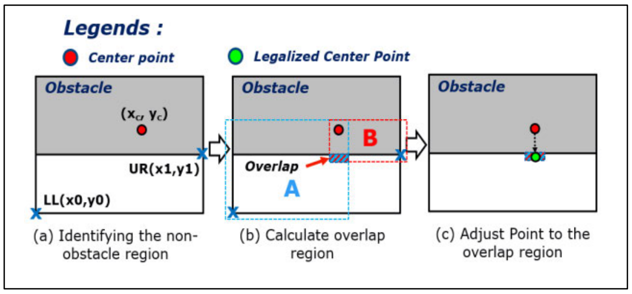

FIGURE 20. The proposed legalization algorithm using Boolean operation on polygons.

the chip as shown in Fig. **21b**. The B2B center point will be auto adjusted to the legal region to reduce global skew.

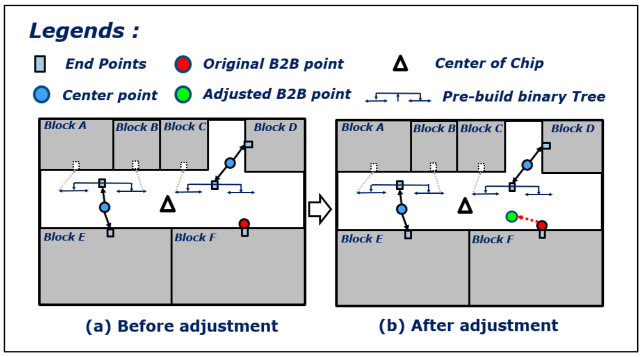

FIGURE 21. Adjustment of back-to-back point towards center of chip to reduce skew.

The algorithm to find the correct k value and the respective center point set $\{C_1(x_1,y_1),\ldots,C_k(x_k,y_k)\} \subseteq \mathbb{R}^d$ for a set of input end points $P(x,y)$ is modeled in Algorithm **2** in Fig. **22**. Basically, based on predefined minimum and maximum values of variables k and E seeds, it will run Algorithm **1** with different combinational the variables values and then compare the returned score S to select the best recipe.

A mathematic approach is used to assign the initial center points c_i of the k-mean algorithm. To improve distribution coverage, floorplan will be partitioned into windows, in which one of the end points enclosed within each window could be assigned as an initial center point. The number of searching windows S_w, as illustrated in Fig. **23a**, is defined as:

$$S_w = [\mathrm{ceil}\left(\sqrt{k}\right)]^2 \tag{3}$$

where $\mathrm{ceil}\left(\sqrt{k}\right)$ is the least integer greater than or equal to $\sqrt{k}$; $S_w > k$

E_y and E_x are basically Y-axis and X-axis offset values of the searching windows. They are used to shift the searching window as illustrated in Fig. **23b** and Fig. **23c**. One initial center point is assigned per window by default. If default method is not enough to fully cover k initial points, random points will be selected to cover the remainders.

The proposed global CTS algorithm, which involves searching for multiple stages of center points to form a clock distribution tree, is modeled in Algorithm **3** in Fig. **24**.

Different blocks may have different clock latencies which mainly depend on their floorplan size, sequential logic counts, and complexity of the clock control logics. For a complex SoC, it is not feasible to perform global clock synthesis with considering the massive number of clock end points within blocks. Thus, in the proposed global CTS algorithm, blocks will be treated as black boxes and input clock pins to the blocks are set as initial end points. Then, for each initial end point, it will be driven by a pre-inserted anchor buffer. These anchor buffer will be treated as new end points which will be pre-adjusted as shown in Fig. **25** to offset clock

```
Algorithm: 2
Inputs:
  End point set, $P(x,y)= \{p_1(x_1,y_1), p_2(x_2,y_2), \ldots\} \subseteq \mathbb{R}^d$
  Minimum number of centers, $k_{min} \in \mathbb{Z}$ where $k > 0$
  Maximum number of centers, $k_{max} \in \mathbb{Z}$ where $k < |P(x,y)|$
  Minimum value of Seed at X-axis, $E_{xmin}$ where $0 \leq E_{xmin} \leq 1$
  Maximum value of Seed at X-axis, $E_{xmax}$ where $0 \leq E_{xmax} \leq 1$
  Granularity of Seed Value at X-axis, $E_{xgra}$ where $0 \leq E_{xgra} \leq 1$
  Minimum value of Seed at Y-axis, $E_{ymin}$ where $0 \leq E_{ymin} \leq 1$
  Maximum value of Seed at Y-axis, $E_{ymax}$ where $0 \leq E_{ymax} \leq 1$
  Granularity of Seed Value at Y-axis, $E_{ygra}$ where $0 \leq E_{ygra} \leq 1$
Outputs:
  Clusters Sets $P_i \subseteq P(x,y)$ where $i = 1$ to $k$
  Cluster center point Set, $C(x,y)= \{c_1(x_{c1}, y_{c1}), \ldots, c_k(x_{ck}, y_{ck})\} \subseteq \mathbb{R}^d$

Set $S_b \in \mathbb{R}$ to a large number.
For $k = k_{min}$ to $k_{max}$ do
  For $E_y = E_{ymin}$ to $E_{ymax}$ do
    For $E_x = E_{xmin}$ to $E_{xmax}$ do
      Define the initial center points set $c_i$ (see Fig. 23)
      Run $k$-mean Algorithm 1 (see Fig. 19) with $P(x,y)$, $k$, and $c_i$
      If $S$ returned is less than $S_b$ then
        Update $S_b$ with value of $S$
        Record the clusters $P_i$
        Record the center point Set, $C(x,y)$
      End if
      Increase $E_x$ by $E_{xgra}$
    End for
    Increase $E_y$ by $E_{ygra}$
  End for
  Increase $k$ by 1
End for
```

FIGURE 22. The algorithm used to search for the best scored center point set for single stage by varying the value of *k* and seed *E*.

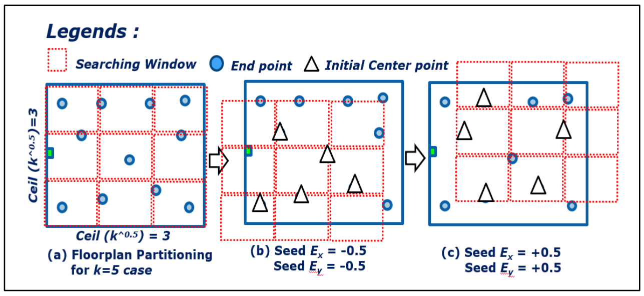
Legends :
Searching Window
End point
Initial Center point
Ceil (k^0.5)=3
Ceil (k^0.5) = 3
(a) Floorplan Partitioning for k=5 case
(b) Seed Ex = -0.5 Seed Ey = -0.5
(c) Seed Ex = +0.5 Seed Ey = +0.5

FIGURE 23. The algorithm to assign the initial center points based on seeds.

latencies within the blocks according to a prebuilt latency per distance Lookup table. In certain special case, multiple clock pins, which are placed close to each other, are created on a block for the same signal. This will cause k-mean algorithm to have an inaccurate clustering result. Thus, anchor buffer should be used to combine these multiple pins into single end point up front.

Algorithm 3 will define the values of n_{max}, k_{min}, k_{max}, E_{gra}, E_{min}, and E_{max} using exact algorithm that extracted from SMEs' experiences in previous projects and training

```
Algorithm: 3
Inputs:
    Initial end point set, $P(x,y)= \{p_1(x_1,y_1), p_2(x_2,y_2),\ldots\} \subseteq \mathbb{R}^d$
Outputs:
    For each stage index $n \in \mathbb{Z}$:
        End point set $P_n(x,y) \subseteq \mathbb{R}^d$ of stage $n$
        Clusters Sets $P_{ni} \subseteq P_n(x,y)$ where $i = 1$ to $k_n$
        Cluster center point Set, $C_n(x_n,y_n) \subseteq \mathbb{R}^d$
Initialization of a floorplan layout database
Adjust end points location based on block level Latency (see Fig. 25)
Define max stage count, $n_{max}$ based on $|P_n(x,y)|$ (see Sub-Section C)
set stage index $n = 1$
set current end point set, $P_n(x,y) = P(x,y)$
repeat
    Define $k_{min}$ and $k_{max}$ (see Sub-Section C)
    Define $E_{gra}$, $E_{min}$ and $E_{max}$ for both X and Y axis (see Sub-Section C)
    Run algorithm 2 (see Fig. 22) with $P_n(x,y)$, $k_{min}$, $k_{max}$, $E_{min}$, $E_{max}$, $E_{gra}$
    Saved returned $C(x,y)$ as $C_n(x,y)$
    Convert $C_n(x,y)$ to become new $P_n(x,y)$ for next stage
    Increase $n$ by 1
until the numbers of returned centers $|C(x,y)|$=1 or $n_{max}$ is met
```

FIGURE 24. **The proposed global CTS algorithm.**

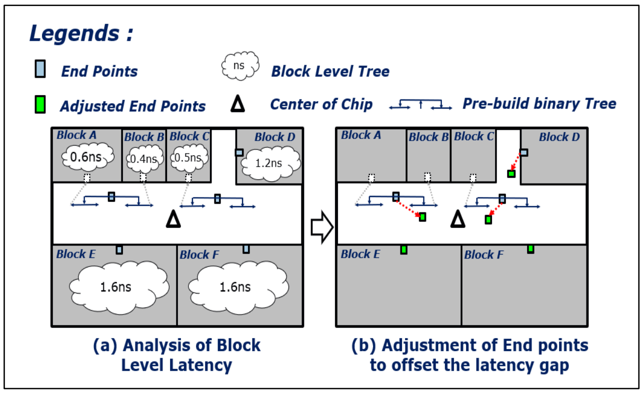

FIGURE 25. **Adjustment of the initial end points towards center of chip to offset the clock latency differences in block level.**

experiments which will be briefly explained in Sub-section **C**. Basically, values of k_{min} and k_{max} are defined based on the number of end points $|P_n(x,y)|$ while E_{gra}, E_{min}, and E_{max} are based on distribution of $P_n(x,y)$. Then, based on the current end point set, the global CTS algorithm will run Algorithm **2** to search for center point set with the best score and convert them to become end points for the next stage. The process repeats until only one point remains in current center point set or the pre-defined maximum stage, which is meant to avoid dead lock, is met.

B. FAST HEURISTIC FLOW

Industry CAD tool is used to heuristically implement the synthesized global clock tree and measure the quality of results. Implementation includes insertion of clock cells at the center points, connecting to the respective end points, repeater insertion, placement, routing and shielding while measurements includes checking the latency, transition time, skew, and jitter. However, detailed physical implementation is taking longer time to complete especially for large and complex SoC. To speed up turn-around-time, which is very important for metaheuristic algorithm to cover larger search-ing space in an acceptable timeframe, a fast heuristic flow is proposed. There are two key components in the flow:

(1) Lookup Table (LUT) of latency per distance for the best know clock repeater recipe, which defines repeater types, metal layers, interval distance, and shielding that produce the global optimized QoR, as covered in our previous GA based HMH flow [43].
(2) A computational geometry technique base fast global-route technique.

Basically, the flow will use a fast-global route technique to estimate Manhattans distance of route that avoid obstacles between two points. Then it calculates the latency of the path by referring to the LUT abovementioned. The proposed fast global-route technique uses Boolean operation on polygons, which is a computational geometry technique [64], as illus-trated in Fig. **26**. First, it converts input data, including the two center points, non-obstacle and obstacle regions into polygons. After that, for each polygon point of the obstacle regions, two virtual rectangular boxes, i.e. A and B as shown in Fig. **26a** and **26b** will be formed. The LL and UR points of the boxes are expanded by $\Delta = 1\mu m$ as in formula **(2)** for overlap calculation. After obtaining the overlap polygons between the boxes with the non-obstacle (θ), the overlaps regions will be geometry Boolean OR to form a routing path as in Fig. **26c**. The Manhattans distance of the path is estimated by dividing the perimeter of the routing path polygons by 2 as shown in Fig. **26d**.

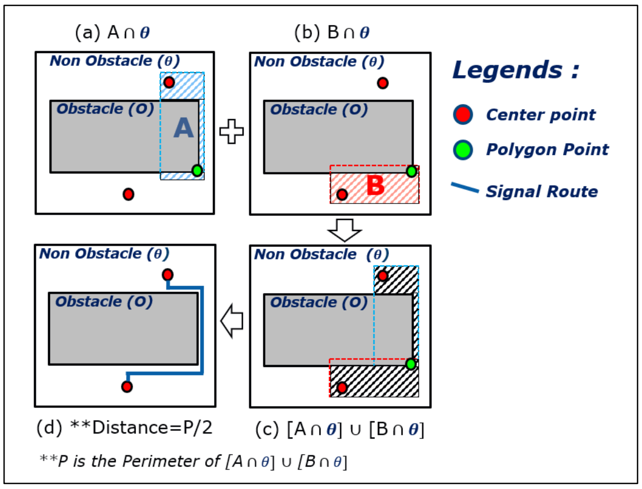

FIGURE 26. **The proposed computational geometry technique based fast global-route technique.**

By performing the geometry computation as in Fig. **26** for all polygon points of the obstacle and then comparing perimeters of the output polygons, we can quickly determine the shortest routing path for two points as in Fig. **27**.

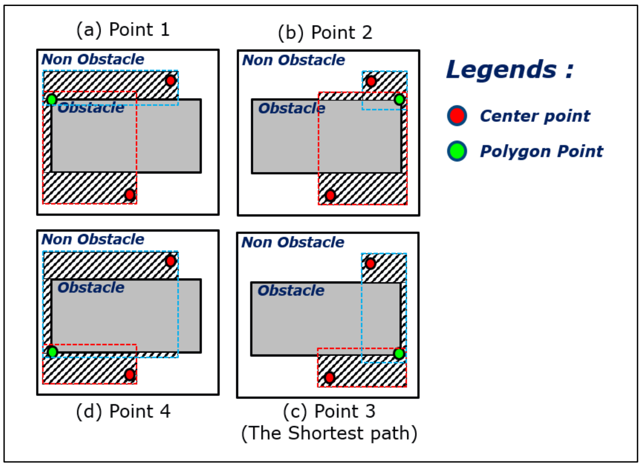

FIGURE 27. The proposed shortest path finding procedure.

To speed up further the global CTS flow, Euclidean distance (Fig. **28a**) and simple Manhattans distance (Fig. **28b**), which are not obstacle aware, can be used. However, compare to an obstacle aware Manhattans distance (Fig. **28c**), these distances are less accurate for the clock QoR estimation, especially for design with larger obstacles.

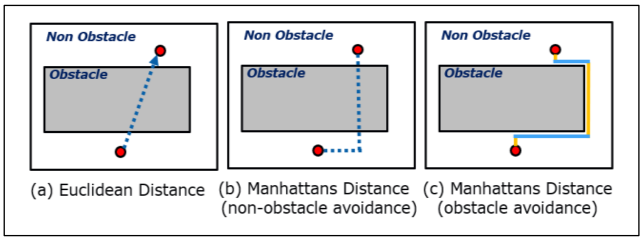
Non Obstacle
Obstacle
(a) Euclidean Distance
(b) Manhattans Distance (non-obstacle avoidance)
(c) Manhattans Distance (obstacle avoidance)

FIGURE 28. Different distance estimation techniques.

C. EXACT ALGORITHM FOR DEAD LOCK SOLVING AND SELECTION

Mathematic approach based exact algorithms [61] are used to model the SME solution selection process, to avoid death lock and solve multi-objectives optimization problem. These algorithms have been fine-tuned based on learning from previous projects and experiments. The following are the key formula used in the proposed global CTS algorithm.

(1) Maximum stage count n_{max}:

$$n_{max} = \text{ceil}\left(\frac{log_{10}|P_i|}{log_{10}2}\right) + \Delta n \tag{4}$$

where initial end point set, $P_i = \{p_1(x_1,y_1), p_2(x_2,y_2),\ldots\} \subseteq \mathbb{R}^d$ and $\Delta n \in \mathbb{Z}$ is user configurable integer for extra stages if needed. ceil( ) is operation of "the least integer greater than or equal to"

Equation **(4)** is derived from $|P_i| = 2^{n_{max}}$ in which the number of times $|P_i|$ could be divided by 2. n_{max} is used to control the stage count which directly impact the total clock jitter and latency.

(2) Minimum (k_{min}) and maximum (k_{max}) of center points:

$$k_{min} = ceil\left(\frac{|P|}{4}\right) + \Delta k_{min};\ k_{min} \in \mathbb{Z};\ 1 \leq k_{min} < |P| \tag{5}$$

$$k_{max} = ceil\left(\frac{|P|}{2}\right) + \Delta k_{max};\ k_{max} \in \mathbb{Z};\ k_{min} \leq k_{max} \leq |P| \tag{6}$$

where current end point set, $P = \{p_1(x_1,y_1), p_2(x_2,y_2),\ldots\} \subseteq \mathbb{R}^d$ and $\Delta k_{min} \in \mathbb{Z}$ and $\Delta k_{max} \in \mathbb{Z}$ are user configurable integers. k_{min} and k_{max} are used to avoid dead lock as well as reduce searching space and turn -around-time.

(3) E_{min}, E_{max}, and E_{gra} for both X and Y-axis:

$$E_{xmin} = \frac{mean(P_x) - C_x}{w_x};\ E_{xmin} \in \mathbb{R} \tag{7}$$

$$E_{xmax} = E_{xmin} + 1;\ E_{xmax} \in \mathbb{R} \tag{8}$$

where chip center point at X-axis, $c_x = \frac{chipwidth}{2} + origin_x$ Mean of end points at X-axis, $mean(P_x) = \frac{1}{|P|}\sum p_x$ window width size, $w_x = \frac{chipwidth}{ceil(\sqrt{k})}$

$$E_{xgra} = 0.25;\ E_{xgra} \in \mathbb{R} \tag{9}$$

Similar formula **(7)**, **(8)**, and **(9)** are used for Y-axis. E_{min}, E_{max}, and E_{gra} are meant to offset the windows that select the initial center points according to the distribution of the end points. E_{gra}can be more accurately estimated with standard deviation value of the *P,* but in this article, to simplify the calculation, we have set it to a fixed 0.25 value.

Different SoCs may have different priorities on PPA. To solve this multi-objective problem, a cost based mathematic algorithm, which is like our previous work [43], is used for rank-based selection. We can use a simple method to aggregate all the criteria into one criterion using a weighted summation [65], as in formula **(10)**.

$$CostS_c = m_d D + m_l L + m_g G + m_p P + m_j J + m_b B \tag{10}$$

where D is the total latency cost; L is the total local skew cost; G is the global skew cost; P is the power consumption cost; J is the jitter cost; B is the back-to-back point cost. m_d, m_l, m_g, m_p, m_j and m_b are user defined weightages as per the SoC priority with all $m>0;\ m \in \mathbb{R}$.

Cost in this article is an aggregated value which is meant for relative comparison only. The formulas of calculation used in the work for each cost are explained below based on a simple example illustrated in Fig. **29**.

Latency cost is the sum of all the Manhattan distances from center points to end points, which is:

$$\text{Latency Cost, } D = \overrightarrow{C_1E_1} + \overrightarrow{C_1E_2} + \overrightarrow{C_2E_3} + \overrightarrow{C_2E_4} + \overrightarrow{C_3E_5}$$

Local skew is referred to the latency gap between the longest and the shortest center-to-endpoint paths, which is

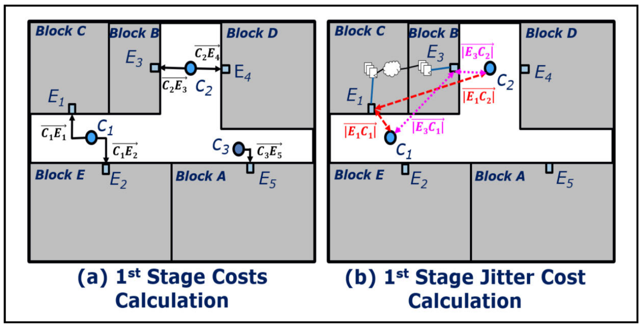
Block C
Block B
Block D
Block E
Block A
E_1
E_2
E_3
E_4
E_5
C_1
C_2
C_3
$\overrightarrow{C_1E_1}$
$\overrightarrow{C_1E_2}$
$\overrightarrow{C_2E_3}$
$\overrightarrow{C_2E_4}$
$\overrightarrow{C_3E_5}$
$|E_1C_1|$
$|E_3C_1|$
$|E_3C_2|$
$|E_1C_2|$
(a) 1st Stage Costs Calculation
(b) 1st Stage Jitter Cost Calculation

FIGURE 29. Examples of costs calculation for 1st stage.

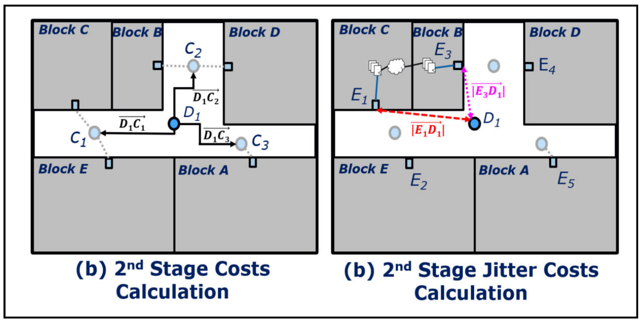
Block C
Block B
Block D
Block E
Block A
C_1
C_2
C_3
D_1
E_1
E_2
E_3
E_4
E_5
$\overrightarrow{D_1C_1}$
$\overrightarrow{D_1C_2}$
$\overrightarrow{D_1C_3}$
$|E_1D_1|$
$|E_3D_1|$
(b) 2nd Stage Costs Calculation
(b) 2nd Stage Jitter Costs Calculation

FIGURE 30. Examples of costs calculation for 2nd stage.

driven by the same center point. Total local skew cost is the sum of local skews of each center point. In this example:

$$\begin{aligned}&\text{Total Local skew Cost, } L\\&\quad= [Max(\overrightarrow{C_1E_1},\overrightarrow{C_1E_2}) - Min(\overrightarrow{C_1E_1},\overrightarrow{C_1E_2})]\\&\qquad+[Max(\overrightarrow{C_2E_3},\overrightarrow{C_2E_4}) - Min(\overrightarrow{C_2E_3},\overrightarrow{C_2E_4})]\end{aligned}$$

Global skew is referred to the latency gap between the longest and the shortest of all center-to-endpoint paths, regardless of center points. In this example:

$$\begin{aligned}\text{Global Skew Cost, } G = {}&[Max(\overrightarrow{C_1E_1},\overrightarrow{C_1E_2},\overrightarrow{C_2E_3},\overrightarrow{C_2E_3},\overrightarrow{C_3E_5})]\\&-[Min(\overrightarrow{C_1E_1},\overrightarrow{C_1E_2},\overrightarrow{C_2E_3},\overrightarrow{C_2E_3},\overrightarrow{C_3E_5})]\end{aligned}$$

Local and global skew will be minimum in the case if all end points fall under back-to-back (B2B) case, in which every end point has its own dedicated center point. To avoid bias in the selection algorithm, B2B point cost is added. In this example:

$$\begin{aligned}\textit{Back-to-back point Cost}, B &= \textit{number of point} * 1000\\&= 1 * 1000\end{aligned}$$

Jitter cost is calculated by first, summing up the Euclidean distances of center-to-endpoint paths of two end points that have timing relationship, and then add on the Euclidean distances of each endpoint to its counterpart center, even they have no direct connection. For the example in Fig. **29b**:

With end point pair, (E_1, E_3) that have timing relationship

$$\textit{Jitter Cost}, J = K_{BC}[\overrightarrow{|E_1C_1|}+\overrightarrow{|E_2C_2|}+\overrightarrow{|E_3C_2|}+\overrightarrow{|E_3C_1|}]$$

where K_{BC} is the designer defined timing criticalness weightage for relationship of timing path between blocks B and C.

Similar formulas are used in the 2nd stage and so on. For a 2nd stage example in Fig. **30b**:

$$\text{Latency Cost, } D = \overrightarrow{D_1C_1} + \overrightarrow{D_1C_2} + \overrightarrow{D_1C_3}$$

Since there is only one center point in this example, global skew cost, G will be equal to the total local skew cost, L and there will be no B2B cost added.

$$\begin{aligned}&\text{Total Local skew Cost, } L\\&\quad= [Max(\overrightarrow{D_1C_1},\overrightarrow{D_1C_2},\overrightarrow{D_1C_3}) - Min(\overrightarrow{D_1C_1},\overrightarrow{D_1C_2},\overrightarrow{D_1C_3})]\end{aligned}$$

To calculate the jitter cost of the 2nd stage onwards, similar formula which sum up the Euclidean distances for endpoint pair that has timing relationship will be used. However, additional script is needed to trace the connectivity to identify the correct center for calculation. In this example:

$$\textit{JitterCost}, J = K_{BC}[\overrightarrow{|E_1D_1|}+\overrightarrow{|E_3D_1|}]$$

V. RESULTS AND DISCUSSION

A 10nm complex SoC, which consists of 132 blocks with more than 600millions gates and 17k memory macros, is used for result discussion. It has total 108 clock domains with 56 of them need to be balanced across the chip. The highest system clock frequency is 1.2GHz. The largest clock domain which distributes clock signal to 4.25million sink points is running at 800MHz. The SoC consists of many MIBs and multi-VDD blocks as shown in Fig. **5** and these special blocks are nonsystematic obstacles to clock distribution as shown in Fig. **6**. The chip is 142mm^2 in die size. Due to the large die size, more than 500k flop repeaters have been inserted into different none-obstacle blocks to meet setup time for different clock domains. To stay competitive in the market, the design of the SoC is planned to complete within 1.5 years. Schedule and performance are the highest priority for this SoC. Thus, area and power consumption have been reprioritized in the analysis.

The complex SoC has two design versions, namely A0 and B0. In A0 version, clock trees are mainly built with a common ASIC technique, which uses state-of-the-Art msCTS engines of a commercial tool as illustrated in Fig **31a**. Basically, an in-house synthesis tool is used to build global clock trunk from PLL to DOPs in non-obstacle blocks. Then, different latency targets are set for the CAD msCTS engine to build clock trees from DOP trees to clock pins of blocks nearby, on top of sequential logics within the non-obstacle blocks. The latency target values are pre-extracted from CTS results of all blocks and block level CTS are performed with CCD fully enabled including to the IO flops. This is a common ASIC technique which focuses on fast-turn-around time. However, with the emerging challenges of a complex SoC as described in Section **III**, the clock QoR is hardly controllable. Consequently, significant amount of effort and

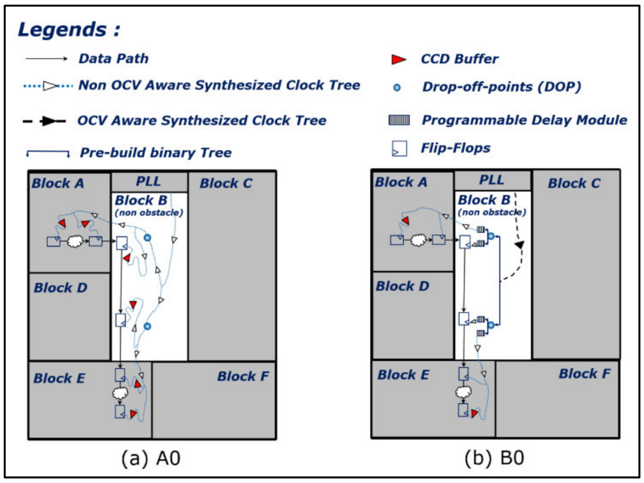

FIGURE 31. Comparison of the clock distribution approaches used in A0 and B0.

time is spent to patch the FC clock design before it was barely acceptable for timing convergence.

As mitigation plan, the proposed FC clock distribution techniques as illustrated in Fig. **31b** have been used in B0. Global clock trunk is built with both BCS and UCS, whereby BCS is used by default but UCS is used on clock domain that must drive large block. An example of the global clock tree in B0 is shown in Fig. **32**. On top of that, ZZB and PDM are also used for fast clock balancing. With that approach deployed on all the 56 clocks in the SoC, acceptable FC clock design has been achieved within five weeks, saved nine weeks compared to A0. The comparison of the FC clock balancing duration for 5 of the key clock domains is shown in Table **5**.

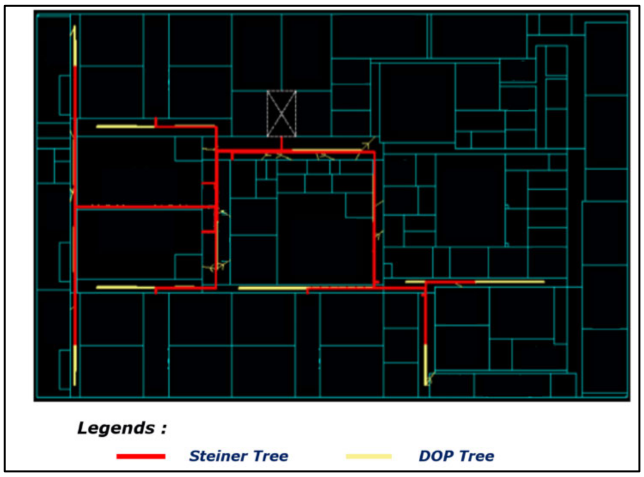

FIGURE 32. An example of a global clock tree implemented with the proposed FC clock distribution design techniques.

Note that due to process technology confidentiality, total clock latency will not be shared in this article. Besides, commercial tools used in the work will not be disclosed.

Comparison of FC clock skew for key clock domains, which include global clock trunk, block level tree, and balancing modules, is shown in in Table **6.** We see averagely

TABLE 5. Comparison of Clock Balancing Duration Between A0 and B0.

Clock Domains	Blocks Count	Total Sinks	Duration		
			A0 (week)	B0 (week)	Delta (%)
CLK_A	50	4252184	14	5	-64.3
CLK_B	17	4075819	8	3	-62.5
CLK_C	21	3323599	9	3	-66.7
CLK_D	5	812973	8	3	-62.5
CLK_E	25	804516	12	4	-66.7
Average					-64.5

16.98% of FC skew improvement in B0 compare to A0 in this SoC. The improvement is mainly due to better controllability of delay tweaking with the introduction of PDM and ZZB, as compare to CAD CTS. However, PDM and ZZB will not improve local skew within a block. A few clock domains have worse global skews in B0 due to larger local skews within a few blocks as highlighted in red color in Table 6. These skews are intendedly added to close some critical timing paths as B0 design has more stringent timing requirements to improve yield in mass production. These useful skews are applied mainly on the on-chip clock controller logics.

TABLE 6. FC Level Clock QoR of A0 vs B0.

Clock Domains	Blocks Count	Total Sinks	Average Skew on IO Flops		
			A0 (ps)	B0 (ps)	Delta (%)
CLK_A	50	4252184	308	244	-20.78
CLK_B	17	4075819	318	255	-19.81
CLK_C	21	3323599	346	224	-35.26
CLK_D	5	812973	505	282	-44.16
CLK_E	25	804516	209	243	16.27
CLK_F	2	650080	757	675	-10.83
CLK_G	6	293952	265	157	-40.75
CLK_H	4	236992	281	156	-44.48
CLK_I	22	226058	267	313	17.23
CLK_J	27	94934	151	153	1.32
CLK_K	16	57858	225	149	-33.78
CLK_L	15	24130	248	209	-15.73
CLK_M	10	23551	2159	1591	-26.31
CLK_N	14	19910	124	148	19.35
Average					-16.98

Besides, by disabling CCD on IO flops and set minimum target latency in block level CTS, we can improve clock skew on IO flops significantly. This has also helped to reduce the effort and time in FC balancing phase. For blocks that are driven by one of the main system clocks shown in Table **7**, averagely 45.59% of clock skew improvement has been achieved. However, latency improvement is not obvious in these case studies as skew is set to have higher priority over latency in the CAD CTS algorithm even though the target latency is already set to minimum. Note that a few outlier sink-points, such as on chip clock controller logics that have large intended/useful skew added to solve timing violations, have been filtered from the results.

Skew comparison of the global clock trunk of the key clock domains is shown in Table **8**. These measurements are on trees from PLL to DOPs excluding the branches with UCS.

TABLE 7. Block Level Clock QoR for Main system Clock of A0 vs B0.

Block	Total Sinks	Latency Delta (%)	Average Skew on IO Flops		
			A0 (ps)	B0 (ps)	Delta (%)
BLK A	569	-31.96	130	64	-50.77
BLK B	13788	-6.16	185	124	-32.97
BLK C	87481	36.14	349	114	-67.34
BLK D	13788	21.68	142	141	-0.70
BLK E	28661	-21.61	240	148	-38.33
BLK F	30462	25.12	288	136	-52.78
BLK G	76615	-19.40	415	204	-50.84
BLK H	119845	-34.25	202	106	-47.52
BLK I	180297	-34.84	378	224	-40.74
BLK J	174969	-23.50	306	120	-60.78
BLK K	217828	25.83	256	148	-42.19
BLK L	55403	-2.42	259	140	-45.95
BLK M	126409	-41.10	544	437	-19.67
BLK N	18755	-14.53	264	132	-50
BLK O	455737	-19.16	407	185	-54.55
BLK P	192736	-35.49	356	150	-57.87
BLK Q	244515	-14.29	246	151	-38.62
BLK R	248760	-6.17	246	160	-34.96
BLK S	84056	22.05	374	127	-66.04
BLK T	33464	-12.54	373	132	-64.61
BLK U	24367	2.07	247	148	-40.08
Average		-8.79			-45.59

TABLE 8. Comparison of SKEW of Global Trunk only for A0 vs B0.

Clock Domains	Block Counts	Global Trunk Skew (From PLL to DOPs)		
		A0 (ps)	B0 (ps)	Delta
CLK A	21	143	40	-103ps (-72%)
CLK B	17	75	31	-44ps (-58%)
CLK C	50	233	45	-188ps (-81%)
CLK D	27	136	42	-94ps (-69%)

With the proposed technique, faster clock design turn-around-time can be achieved and consequently more iterations of skew fine-tuning can be performed. Thus, significant skew improvement on BCS clock trunk has been achieved in B0.

In A0, global clock tree is CTS without OCV aware. In B0, the proposed OCV aware BCS algorithm is used for the global clock trunk by default and for clock domains with large blocks, OCV aware UCS algorithm is used to reduce divergence effectively. An example of non OVC aware clock tree vs OVC aware is shown in Fig. **33**. Comparison of diverged repeater stage between A0 vs B0 for the critical clock domains is shown in Table **9**. We believe the improvement is vary by design. In these critical clock domains, we can achieve averagely up to 42.75% reduction of diverged repeater stages. The magnitude of improvement is significant due to the large SoC size which requires more stages of clock repeaters and non-systematic placement of IP blocks.

The proposed HMH global CTS flow can operate with different route distance estimation techniques, which trade off accuracy for runtime, as described in Section **IVB.** To evaluate the trade off, smaller SoC testcases (TC) with different die sizes have been used. Snapshot of a k-mean algorithm built global tree from one of the TCs is shown in Fig. **34**. It shows how a simple global clock tree is built with three stages of centroids with the proposed fast global route technique. Runtime and skew comparison between non-obstacle Manhattans (NoM), fast global route (FGR) and actual global route (AGR) are shown in Table **10**. For apple-to-apple comparison, experiments were performed using the same pre-built floorplans, CAD tool version and machine resources specification for each TC.

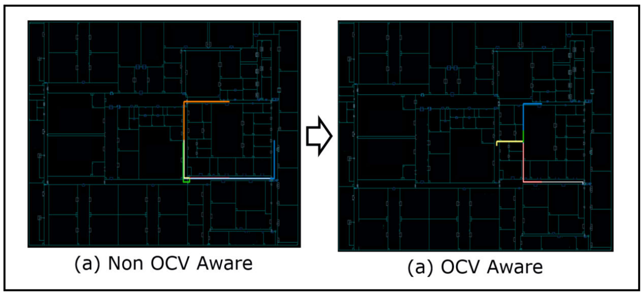

FIGURE 33. An example of non OCV aware vs OCV aware global tree.

TABLE 9. Comparison of Divergence Stages for A0 vs B0 Scheme.

Clocks	Maximum Diverged Repeater Stages		
	A0 (Non OCV Aware)	B0 (OCV Aware)	Delta (%)
CLK A	36	20	-44
CLK B	28	14	-50
CLK C	50	30	-40
CLK D	48	30	-37
Average			-42.75

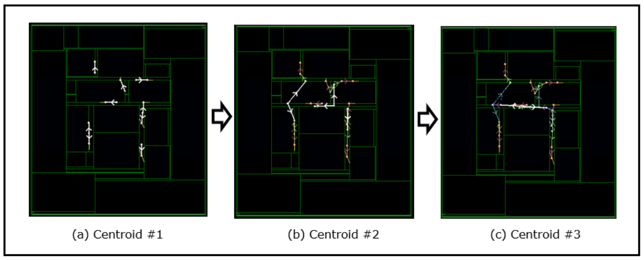

FIGURE 34. A snapshot of k-mean algorithm built global clock tree.

TABLE 10. Run time comparison between different distance estimation techniques.

SOC	Die Size (mm^2)	Block count	Machine Run Time (Hour)			Skew (ps)		
			NoM	FGR	AGR	NoM	FGR	AGR
TC1	31.7	11	0.4	1	36	150.5	55	54.3
TC2	44.7	19	0.58	3.33	54	143.7	61.3	60.4
TC3	134	35	1.25	7	288	189.6	122.7	127.9

From these case studies, we noticed that runtime of the HMH flow increases as die size and block count increase.

Among the techniques, NoM has the fastest turn-around-time but its skew estimation is not accurate especially if any signal has accidentally feedthrough large obstacle blocks. By using FGR or AGR, obstacles can be avoided as shown on one of the runs of TC2 in Fig. **35**. FGR and AGR produce design with close skew results as they are used just to estimate the route distance, in which obstacles are avoided, to help the exact algorithm in selection process. The selection process is basically a relative comparison algorithm, thus minor inaccuracy of FGR will not change the final decision in most of the time. However, with the proposed FGR technique, run time of the proposed HMH flow can be reduced up to 41.1 times for the larger TC3 if compare to the AGR which route with CAD tool. We believe this is because AGR consider more variables such as every possible routing track available [66] while the proposed FGR considers only the corner points of obstacles.

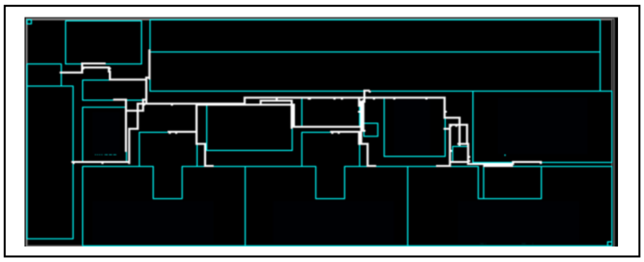
FIGURE 35. A snapshot of obstacle aware global clock synthesis.

Similar to the analysis method used by researchers [69], we have further evaluated the effectiveness the proposed approach by comparing to a state-of-the-Art CTS of a commercial tool. A smaller hierarchical design, which is built based on 7nm technology node, have been used in the experiment. The design consists of multiple instantiated IP blocks as well as flattened standard cells at top level. There are 2 system clock domains in the design and the highest frequency is running at 1GHz. The clock trees have been built with both the proposed approach (PA) and a state-of-the-Art CTS of a commercial tool. Analysis is performed using the static timing analysis and clock QoR reporting engines of the same tool. Fig. **36** shows the clock tree snapshots of the experiment and Fig. **37** shows the normalized skew results. In this experiment, we achieved 28.89% better global skew with PA compare to the CTS.

Clock re-convergence pessimism (CRP) is a difference in delay along the common part of the launching and capturing clock paths which models the impact of variation on skew that causes inaccuracy in timing [53] while clock re-convergence pessimism removal (CRPR) is an automated correction of this inaccuracy in timing calculation which is available in most of the design tools. Higher CRPR value in a timing path directly indicates lower OCV induced jitter effects on the clock distribution network of the path. Therefore, to measure the jitter reduction effectiveness, we have calculated average CRPR values of 100 thousand worst timing paths, which excluded block level internal timing paths, for each clock domains. The result in Table **11** shows that PA produces averagely 31.15ps less jitter, which is 3.115% of 1GHz clock period, compare to the state-of-the-Art CTS of the tool in this experiment.

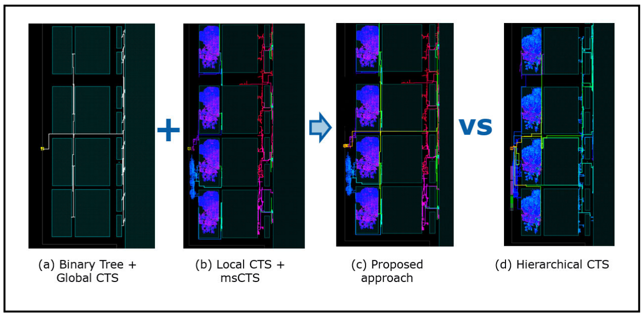

FIGURE 36. A snapshot the trees built with the proposed approach vs a state-of-the-art hierarchical CTS.

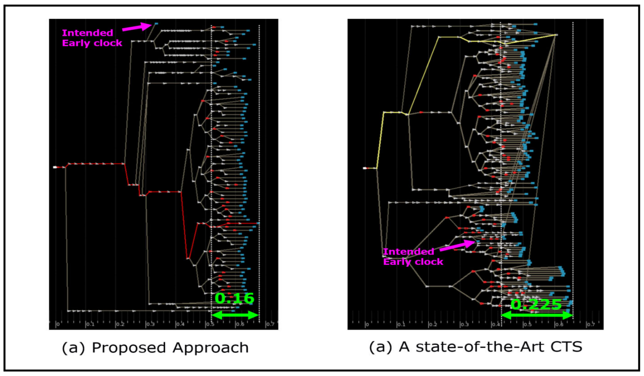

FIGURE 37. Comparison of normalized clock latency and global skew between the proposed approach versus a state-of-the-Art CTS.

TABLE 11. Comparison of CRPR values of PA vs CTS.

Clock Domains	*IP Block Clock Inputs*	*Clock Sinks*	*CRPR*			
			PA (ps)	*CTS (ps)*	*Delta (ps)*	*Delta (%) Over 1GHz Clock Period*
CLK N	*24*	*201945*	*53.5*	*22.7*	*30.8*	*3.08*
CLK S	*24*	*199509*	*52.9*	*21.4*	*31.5*	*3.15*
AVERAGE			*53.2*	*22.05*	*31.15*	*3.115*

VI. CONCLUSION

On top of tighter design requirements due to many higher frequency clock domains, there are emerging design challenges of a complex SoC, such as none systematically placed obstacles as well as repeater flops, CCD features, and high OCV effects due to sub-nano technology. Furthermore, to stay competitive in the current dynamic market environment, it is a necessity for a complex SoC to be designed with high productivity and low development cost.

In this article, we present a practical full chip clock design technique that solves the emerging challenges of a complex SoC. It includes a flexible topology and flow that not only solves the new design complexities but aligns with actual SoC design milestones requirements. Besides, an HMH algorithm is introduced to overcome CAD capacity and runtime issues for a complex SoC and subsequently enable fast obstacle and OCV aware global clock synthesis. With these techniques, the team managed to achieve better QoR with averagely 64.5% shorter time to clock frozen milestone compare to the prior SoC which uses conventional ASIC technique.

Besides, we are pursuing extensions of the hybrid meta-heuristic technique into multi-hierarchical clock design optimization, which allows global and block level HMH algorithms to collaborate automatically using Particle Swarm Optimization (PSO) algorithm [67].

ACKNOWLEDGMENT

Eng Keong would like to thank Navin Saxena, Mathew Moore, Phil Welling, Liew Yin Hao, T. W. Tan, James Lim, Wen Chin Lim, Roberto Perez, Jugantor Chetia, Kah Weng Ng, Boon Phin Tan, Pik Lay Teo, and people whom had helped or shared helpful information with him on the artificial intelligent algorithm as well as design methodologies.

REFERENCES

[1] K. D. Boese and A. B. Kahng, "Zero-skew clock routing trees with minimum wirelength," in *Proc. 5th Annu. IEEE Int. ASIC Conf. Exhib.*, Sep. 1992, pp. 17–21, doi: 10.1109/ASIC.1992.270316.

[2] S. Pullela, N. Menezes, J. Omar, and L. T. Pillage, "Skew and delay optimization for reliable buffered clock trees," in *Proc. Int. Conf. Comput. Aided Design (ICCAD)*, Nov. 1993, pp. 556–562, doi: 10.1109/ICCAD.1993.580114.

[3] J. L. Neves and E. G. Friedman, "Design methodology for synthesizing clock distribution networks exploiting nonzero localized clock skew," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 4, no. 2, pp. 286–291, Jun. 1996, doi: 10.1109/92.502201.

[4] M. B. Maaz and M. A. Bayoumi, "A non-zero clock skew scheduling algorithm for high speed clock distribution network," in *Proc. IEEE Int. Symp. Circuits Syst. VLSI (ISCAS)*, May 1999, pp. 382–385, doi: 10.1109/ISCAS.1999.780175.

[5] H. Fair and D. Bailey, "Clocking design and analysis for a 600 MHz alpha microprocessor," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 1998, pp. 398–399, doi: 10.1109/ISSCC.1998.672551.

[6] P. J. Restle, T. G. McNamara, D. A. Webber, P. J. Camporese, K. F. Eng, K. A. Jenkins, D. H. Allen, M. J. Rohn, M. P. Quaranta, D. W. Boerstler, C. J. Alpert, C. A. Carter, R. N. Bailey, J. G. Petrovic, B. L. Krauter, and B. D. McCredie, "A clock distribution network for microprocessors," in *Proc. Symp. VLSI Circuits. Dig. Tech. Papers*, May 2000, pp. 184–187, doi: 10.1109/VLSIC.2000.852885.

[7] M. Mori, H. Chen, B. Yao, and C.-K. Cheng, "A mulitple level network approach for clock skew minimization with process variations," in *Proc. ASP-DAC Asia South Pacific Design Autom. Conf.*, Jan. 2004, pp. 263–268, doi: 10.1109/ASPDAC.2004.1337577.

[8] Q. Gu and J. Gu Brianli, "Implement 'mesh+local trees' clock design flow in encounter," in *Proc. 9th Int. Conf. Solid-State Integrated-Circuit Technol.*, Oct. 2008, pp. 2288–2291, doi: 10.1109/ICSICT.2008.4735027.

[9] A. L. Sobczyk, A. W. Luczyk, and W. A. Pleskacz, "Power dissipation in basic global clock distribution networks," in *Proc. IEEE Design Diag. Electron. Circuits Syst.*, Apr. 2007, pp. 1–4, doi: 10.1109/DDECS.2007.4295287.

[10] S. M. Reddy, G. R. Wilke, and R. Murgai, "Analyzing timing uncertainty in mesh-based clock architectures," in *Proc. Design Autom. Test Eur. Conf.*, Mar. 2006, pp. 1–6, doi: 10.1109/DATE.2006.243962.

[11] F. H. A. Asgari and M. Sachdev, "A low-power reduced swing global clocking methodology," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 12, no. 5, pp. 538–545, May 2004, doi: 10.1109/TVLSI.2004.826204.

[12] J. Reuben, A. Anuroop, and H. M. Kittur, "Clock frequency doubler circuit for multiple frequencies and its application in a CDN to reduce power," in *Proc. Int. Conf. Comput., Electron. Electr. Technol. (ICCEET)*, Mar. 2012, pp. 752–756, doi: 10.1109/ICCEET.2012.6203744.

[13] K. Takeuchi, A. Yoshikawa, M. Komoda, K. Kotani, H. Matsushita, Y. Katsuki, Y. Yamamoto, and T. Sato, "Clock-skew test module for exploring reliable clock-distribution under process and global voltage-temperature variations," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 16, no. 11, pp. 1559–1566, Nov. 2008, doi: 10.1109/TVLSI.2008.2000975.

[14] J. Jang, O. Franza, and W. Burleson, "Period jitter estimation in global clock trees," in *Proc. 12th IEEE Workshop Signal Propag. Interconnects*, May 2008, pp. 1–4, doi: 10.1109/SPI.2008.4558367.

[15] R. Wang, T. Okamoto, and C.-K. Cheng, "Symmetrical buffer placement in clock trees for minimal skew immune to global on-chip variations," in *Proc. IEEE Int. Conf. Comput. Design*, Oct. 2009, pp. 23–28, doi: 10.1109/ICCD.2009.5413180.

[16] W. D. Grover, J. Brown, T. Friesen, and S. Marsh, "All-digital multipoint adaptive delay compensation circuit for low skew clock distribution," *Electron. Lett.*, vol. 31, no. 23, pp. 1996–1998, Nov. 1995, doi: 10.1049/el:19951362.

[17] C. E. Dike, N. A. Kurd, P. Patra, and J. Barkatullah, "A design for digital, dynamic clock deskew," in *Proc. Symp. VLSI Circuits. Dig. Tech. Papers*, Mar. 2003, pp. 21–24, doi: 10.1109/VLSIC.2003.1221151.

[18] S. Fairbanks and S. Moore, "Self-timed circuitry for global clocking," in *Proc. 11th IEEE Int. Symp. Asynchronous Circuits Syst.*, Mar. 2005, pp. 86–96, doi: 10.1109/ASYNC.2005.29.

[19] F. Ishihara, C. Klingner, and K. Agawa, "Clock design of 300 MHz 128-bit 2-way superscalar microprocessor," in *Proc. . Design Autom. Conf.*, 2000, pp. 647–652, doi: 10.1109/ASPDAC.2000.835179.

[20] J. Rosenfeld and E. G. Friedman, "Design methodology for global resonant H-Tree clock distribution networks," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2006, p. 4, doi: 10.1109/ISCAS.2006.1693024.

[21] Z. Mohammadi-Arfa and A. Jahanian, "A hybrid RF/metal clock routing algorithm to improve clock delay and routing congestion," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI*, Jul. 2011, pp. 138–143, doi: 10.1109/ISVLSI.2011.87.

[22] Q. Ding and T. Mak, "Hybrid interconnect network for on-chip low-power clock distribution," *Electron. Lett.*, vol. 55, no. 5, pp. 244–246, Mar. 2019, doi: 10.1049/el.2018.6570.

[23] (2015). *The International Technology Roadmap for Semiconductors 2.0.* Edition, Executive Report. [Online]. Available: http://www.itrs2.net/

[24] G. Ellis, L. T. Pileggi, and R. A. Rutenbar, "A hierarchical decomposition methodology for multistage clock circuits," in *Proc. IEEE Int. Conf. Comput. Aided Design (ICCAD)*, Nov. 1997, pp. 266–273, doi: 10.1109/ICCAD.1997.643530.

[25] D.-J. Lee and I. L. Markov, "Obstacle-aware clock-tree shaping during placement," in *Proc. Int. Symp. Phys. Design ISPD*, Mar. 2011, pp. 205–216, doi: 10.1145/1960397.1960425.

[26] Y. Cai, C. Deng, Q. Zhou, H. Yao, F. Niu, and C. N. Sze, "Obstacle-avoiding and slew-constrained clock tree synthesis with efficient buffer insertion," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 1, pp. 142–155, Jan. 2015, doi: 10.1109/TVLSI.2014.2300174.

[27] P. P. Saha, S. Saha, and T. Samanta, "Rectilinear steiner clock tree routing technique with buffer insertion in presence of obstacles," in *Proc. 28th Int. Conf. VLSI Design*, Jan. 2015, pp. 447–451, doi: 10.1109/VLSID.2015.81.

[28] M. Liu, Z. Zhang, W. Sun, and D. Wang, "Obstacle-aware symmetrical clock tree construction," in *Proc. IEEE 60th Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2017, pp. 515–518, doi: 10.1109/MWSCAS.2017.8052973.

[29] (2019). *Clock Jitter Definitions and Measurement Methods, SiT-AN10007 Rev 1.21.* [Online]. Available: http://www.sitime.com/.

[30] A. Chakraborty, P. Sithambaram, K. Duraisami, A. Macii, E. Macii, and M. Poncino, "Thermal resilient bounded-skew clock tree optimization methodology," in *Proc. Design Autom. Test Eur. Conf.*, Mar. 2006, p. 6, doi: 10.1109/DATE.2006.243740.

[31] X.-W. Shih, T.-H. Hsu, H.-C. Lee, Y.-W. Chang, and K.-Y. Chao, "Symmetrical buffered clock-tree synthesis with supply-voltage alignment," in *Proc. 18th Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2013, pp. 447–452, doi: 10.1109/ASPDAC.2013.6509637.

[32] T.-J. Wang, S.-H. Huang, W.-K. Cheng, and Y.-C. Chou, "Top-level activity-driven clock tree synthesis with clock skew variation considered," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2016, pp. 2591–2594, doi: 10.1109/ISCAS.2016.7539123.

[33] H.-Y. Kao, Y. Lee, S.-H. Huang, W.-K. Cheng, and Y.-C. Chou, "An industrial design methodology for the synthesis of OCV-aware top-level clock tree," in *Proc. 6th Int. Symp. Next Gener. Electron. (ISNE)*, May 2017, pp. 1–3, doi: 10.1109/ISNE.2017.7968732.

[34] S. Elassaad, ''Clock driven design planning,'' Elect. Eng. Comput. Sci. Univ. California at Berkeley, Berkeley, CA, USA, Tech. Rep. UCB/EECS-2008-98, 2008. [Online]. Available: http://digitalassets.lib.berkeley.edu/techreports/ucb/text/EECS-2008-98.pdf

[35] C. Yeh, G. Wilke, H. Chen, S. Reddy, H. Nguyen, T. Miyoshi, W. Walker, and R. Murgai, ''Clock distribution architectures: A comparative study,'' in *Proc. 7th Int. Symp. Qual. Electron. Design (ISQED)*, Mar. 2006, p. 7, doi: 10.1109/ISQED.2006.33.

[36] W.-T.-J. Chan, A. B. Kahng, S. Nath, and I. Yamamoto, ''The ITRS MPU and SOC system drivers: Calibration and implications for design-based equivalent scaling in the roadmap,'' in *Proc. IEEE 32nd Int. Conf. Comput. Design (ICCD)*, Oct. 2014, pp. 153–160, doi: 10.1109/ICCD.2014.6974675.

[37] L. Albanese, ''Managing derivative SoC design projects to better results,'' in *Proc. Int. Symp. Signals, Circuits Syst. SCS*, Mar. 2003, pp. 470–477, doi: 10.1109/ISQED.2004.1283718.

[38] Y. Liao, N. Raman, L. Kong, and J.-Y. Chang, ''Efficient floor-planning methodology for the jasper forest SoC on a 45 nanometer process,'' in *Proc. IEEE 8th Int. Conf. ASIC*, Oct. 2009, pp. 407–410, doi: 10.1109/ASICON.2009.5351251.

[39] A. Rajaram and D. Z. Pan, ''Robust chip-level clock tree synthesis,'' *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 30, no. 6, pp. 877–890, Jun. 2011, doi: 10.1109/TCAD.2011.2106852.

[40] K. Han, J. Li, A. B. Kahng, S. Nath, and J. Lee, ''A global-local optimization framework for simultaneous multi-mode multi-corner clock skew variation reduction,'' in *Proc. 52nd Annu. Design Autom. Conf.*, Jun. 2015, pp. 1–6, doi: 10.1145/2744769.2744776.

[41] H. Kodama, M. Mizuno, K. Nose, and A. Tanaka, ''Frequency-hopping Vernier clock generators for multiple clock domain SoCs,'' in *Proc. IEEE Custom Integr. Circuits Conf.*, Oct. 2004, pp. 91–94, doi: 10.1109/CICC.2004.1358744.

[42] R. Lu, G. Zhong, C.-K. Koh, and K.-Y. Chao, ''Flip-flop and repeater insertion for early interconnect planning,'' in *Proc. Design, Autom. Test Eur. Conf. Exhib.*, Mar. 2002, pp. 690–695, doi: 10.1109/DATE.2002.998374.

[43] E. K. Teh, M. A. Md Zawawi, M. F. P. Mohamed, and N. A. M. Isa, ''Practical System-on-Chip repeater design with hybrid meta-heuristic techniques,'' *IEEE Access*, vol. 6, pp. 46334–46345, 2018, doi: 10.1109/ACCESS.2018.2866394.

[44] M. McDermott, *Global Clocking*, document EE 382M Class Notes, The University of Texas at Austin, 2015. [Online]. Available: http://users.ece.utexas.edu/~mcdermot/vlsi1/VLSI2_SP_2017/vlsi2/lectures/12.pdf

[45] D. W. Dobberpuhl *et al.*, ''A 200-MHz 64-b dual-issue CMOS microprocessor,'' *IEEE J. Solid-State Circuits*, vol. 27, no. 11, pp. 1555–1567, Nov. 1992, doi: 10.1109/4.165336.

[46] N. Y. Zhou, P. Restle, J. Palumbo, J. Kozhaya, H. Qian, Z. Li, C. J. Alpert, and C. Sze, ''Pacman: Driving nonuniform clock grid loads for low-skew robust clock network,'' in *Proc. ACM/IEEE Int. Workshop Syst. Level Interconnect Predict. (SLIP)*, San Francisco, CA, USA, 2014, pp. 1–5, doi: 10.1145/2633948.2633953.

[47] J. Jiang, ''Pin allocation for clock routing,'' in *Proc. 2nd Int. Conf. ASIC*, Oct. 1996, pp. 35–38, doi: 10.1109/ICASIC.1996.562744.

[48] V. Bhargava, N. Haider, and N. Sarpotdar, ''IO clock network skew & performance analysis: A Pentium-D case study,'' in *Proc. IEEE Custom Integr. Circuits Conf.*, Sep. 2006, pp. 345–348, doi: 10.1109/CICC.2006.320918.

[49] H.-L. Chen and H.-M. Chen, ''On achieving low-power SoC clock tree synthesis by transition time planning via buffer library study,'' in *Proc. IEEE Int. SOC Conf.*, Sep. 2006, pp. 203–206, doi: 10.1109/SOCC.2006.283881.

[50] S. Gautam, ''Analysis of multi-bit flip flop low power methodology to reduce area and power in physical synthesis and clock tree synthesis in 90nm CMOS technology,'' in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Sep. 2014, pp. 570–574, doi: 10.1109/ICACCI.2014.6968550.

[51] D. Bode, M. Berekovic, A. Borkowski, and L. Buker, ''QoR analysis of automated clock-mesh implementation under OCV consideration,'' in *Proc. 13th Euromicro Conf. Digit. Syst. Design, Archit., Methods Tools*, Sep. 2010, pp. 141–146, doi: 10.1109/DSD.2010.60.

[52] J. Lu and B. Taskin, ''Post-CTS clock skew scheduling with limited delay buffering,'' in *Proc. 52nd IEEE Int. Midwest Symp. Circuits Syst.*, Aug. 2009, pp. 224–227, doi: 10.1109/MWSCAS.2009.5236113.

[53] I. L. Markov and D.-J. Lee, ''Algorithmic tuning of clock trees and derived non-tree structures,'' in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2011, pp. 279–282, doi: 10.1109/ICCAD.2011.6105342.

[54] Y.-Y. Chen, J.-L. Huang, and T. Kuo, ''Implementation of programmable delay lines on off-the-shelf FPGAs,'' in *Proc. IEEE Autotestcon*, Sep. 2013, pp. 1–4, doi: 10.1109/AUTEST.2013.6645040.

[55] L. Kliemann and P. Sanders, ''Theoretical analysis of the k-means algorithem—A survey,'' in *Algorithm Engineering*. Cham, Switzerland: Springer, 2016, ch. 3, pp. 81–116.

[56] W. Chen, C. K. Wang, H. M. Chen, Y. C. Chou, and C. H. Tsai, ''A comparative study on multisource clock network synthesis,'' in *Proc. SASIMI*, 2016, pp. 1–5.

[57] H. I. Osman and G. Laporte, ''Metaheuristics: A bibliography,'' *Ann. Oper. Res.*, vol. 63, pp. 513–623, Oct. 1996.

[58] C. Blum and A. Roli, ''Metaheuristics in combinatorial optimization: Overview and conceptual comparison,'' *ACM Comput. Surveys*, vol. 35, no. 3, pp. 268–308, Sep. 2003.

[59] S. Muthuraman and V. P. Venkatesan, ''A comprehensive study on hybrid meta-heuristic approaches used for solving combinatorial optimization problems,'' in *Proc. World Congr. Comput. Commun. Technol. (WCCCT)*, Feb. 2017, pp. 185–190, doi: 10.1109/WCCCT.2016.53.

[60] G. R. Raidl, ''A unified view on hybrid metaheuristics,'' in *Proc. 3rd Int. Workshop Hybrid Metaheuristics* in Lecture Notes in Computer Science, vol. 4030, Canaria, Spain: Springer, 2006, ch. 1, pp. 1–12.

[61] G. I. Zobolas, C. D. Tarantilis, and G. Ioannou, ''Exact, heuristic and meta-heuristic algorithms for solving shop scheduling problems,'' in *Studies in Computational Intelligence (SCI)*, vol. 128. Berlin, Germany: Springer, 2008, pp. 1–40.

[62] F. Jafarzadehpour, A. S. Molahosseini, A. A. E. Zarandi, and L. Sousa, ''Efficient modular adder designs based on thermometer and one-hot coding,'' *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 9, pp. 2142–2155, Sep. 2019, doi: 10.1109/TVLSI.2019.2919609.

[63] B. A. Trakhtenbrot, ''A survey of Russian approaches to Perebor (brute-force searches) algorithms,'' *Ann. Hist. Comput.*, vol. 6, no. 4, pp. 384–400, Oct. 1984, doi: 10.1109/MAHC.1984.10036.

[64] S. Sengupta, A. Sinharay, and T. Bakshi, ''A computational geometry based cell migration technique for VLSI placement problem,'' in *Proc. IEEE 2nd Int. Conf. Recent Trends Inf. Syst. (ReTIS)*, Jul. 2015, pp. 503–508, doi: 10.1109/ReTIS.2015.7232931.

[65] A. Petrowski, ''Extensions of evolutionary algorithms to multimodal and multiobjective optimization,'' in *Metaheuristics*. Cham, Switzerland: Springer, 2016, ch. 11, p. 314.

[66] H. Chen, ''Global and detailed routing,'' in *Electronic Design Automation: Synthesis, Verification, and Test (Systems on Silicon)*. San Mateo, CA, USA: Morgan Kaufmann, 2009, ch. 12, pp. 687–750.

[67] J. Kennedy and R. Eberhart, ''Particle swarm optimization,'' in *Proc. Int. Conf. Neural Netw. ICNN*, Nov. 1995, pp. 1942–1948, doi: 10.1109/ICNN.1995.488968.

[68] R. Islam and M. R. Guthaus, ''HCDN: hybrid-mode clock distribution networks,'' *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 1, pp. 251–262, Jan. 2019, doi: 10.1109/TCSI.2018.2866224.

[69] K. Han, A. B. Kahng, and J. Li, ''Optimal generalized H-Tree topology and buffering for high-performance and low-power clock distribution,'' *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 2, pp. 478–491, Feb. 2020, doi: 10.1109/TCAD.2018.2889756.

ENG KEONG TEH was born in Malaysia, in 1976. He received the B.Eng. degree (Hons.) in electrical and electronics engineering from Universiti Teknologi Malaysia (UTM), Skudai, Malaysia, in 1999, and the M.Sc. degree in micro-electronics engineering from Multimedia University (MMU), Cyberjaya, Malaysia, in 2010. He is currently pursuing the Ph.D. degree in electrical and electronics engineering with Universiti Sains Malaysia (USM), Nibong Tebal, Malaysia. He has 19 years industrial experiences and is currently a Senior Staff Design Engineer with Intel Microelectronic Malaysia Sdn. Bhd., Malaysia. His current research interests include artificial intelligent and machine learning in complex SoC full chip floor plan, physical integration, and convergence.

MOHAMAD ADZHAR MD ZAWAWI received the B.Eng. degree in electrical and electronic engineering from the University of Sheffield, in 2003, the M.Sc. degree in electronic systems design engineering from Universiti Sains Malaysia, in 2010, and the Ph.D. degree in electrical and electronic engineering from The University of Manchester, in 2015. He currently holds an academic position with the School of Electrical and Electronic Engineering, Universiti Sains Malaysia. His research interests include micro/nanoelectronics fabrication, IC design, modeling and simulation of semiconductor devices, and sub-Thz and Thz MMIC oscillator based on resonant tunneling diodes.

MOHAMED FAUZI PACKEER MOHAMED was born in Kuala Lumpur, Malaysia, in 1978. He received the B.Eng. degree (Hons.) in electrical and electronics engineering from Universiti Tenaga Nasional (UNITEN) Kajang, Selangor, Malaysia, in 2002, the M.Sc. degree in electronics system design engineering from Universiti Sains Malaysia (USM), Nibong Tebal, in 2010, and the Ph.D. degree in electrical and electronics engineering from The University of Manchester (UoM), Manchester, U.K., in 2015. In 2015, he joined the School of Electrical and Electronics Engineering, USM, as a Senior Lecturer. He has seven years industrial experiences back from 2002 to 2009 in semiconductor wafer fabrication and IC packaging prior joining the university as a Lecturer. His current research interests include simulation, design, fabrication, and characterization of high RF and high-power devices based on compound semiconductor materials.

NOR ASHIDI MAT ISA received the B.Eng. degree (Hons.) in electrical and electronic engineering from Universiti Sains Malaysia (USM), in 1999, and the Ph.D. degree in electronic engineering (majoring in image processing and artificial neural network) in 2003. He is currently a Professor and the Deputy Dean (Academic, Students, and Alumni) with the School of Electrical and Electronic Engineering, USM. As of now, he has led his Imaging and Intelligent System Research Team (ISRT) Research Group to publish at both national and international arena. He has published more than 130 manuscripts indexed in ISI out of a total of more than 250 publications. His research interests include intelligent systems, image processing, neural networks, biomedical engineering, and intelligent diagnostic systems and algorithms. Due to his outstanding achievement in research, he gained recognition, both national and internationally. Some of his achievements are appointed as a Visiting Professor by two universities in China, appointed as a Keynote Speaker in five international conferences, invited as a Speaker in four international conferences, appointed as a Ph.D. External Examiner, appointed on editorial board for four international journals, and appointed as a Reviewer of more than 80 ISI indexed journals. He was also appointed as a Mentor of two research grants from Malaysian local universities. As a Research Supervisor, he has successfully graduated 17 Ph.D. and 20 Masters (by research mode) students.