IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 3, 2018, accepted August 9, 2018, date of publication August 22, 2018, date of current version September 7, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2866394

Practical System-on-Chip Repeater Design With Hybrid Meta-Heuristic Techniques

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

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

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

This work was supported by the Intel Microelectronics (M) Sdn. Bhd. under Extended Education Program.

ABSTRACT This paper recommends a practical way to insert buffer and flop repeaters onto global signals of a complex system-on-chip (SoC). With the advent of deep sub-micrometer technology and new business environment, the market prefers a highly integrated SoC with fast design productivity and low development cost. We observed that many algorithms proposed in the prior arts, which used exact algorithms to optimize design solely for performance, power, and/or area, are no longer practical. With that, we introduced a hybrid meta-heuristic based flow, which combines meta-heuristic algorithm with different artificial intelligent algorithms such as exact and heuristic algorithms, to search for a near optimum buffer repeater insertion recipe, optimize the floor-plan pin placement, and then correctly insert flop repeater into the design. Our experiments on 10- and 14-nm SoC products showed that the flow managed to produce a ''good enough'' quality of repeater designs with fast turn-around-time and less design effort.

INDEX TERMS Artificial intelligent, buffer repeater, deep sub-micrometer, flop repeater, hybrid meta-heuristic, system-on-a-chip (SoC).

I. INTRODUCTION

In deep submicron design, the interconnect delay has become one of the dominant performance limiting factors to a circuit. Thus, many interconnect optimization techniques have been introduced including buffer and flop repeaters insertion. For buffer repeater optimization, some researches [1], [2] had introduced the concept of feasible regions or also known as buffer blocks. In previous work [2], the team have also derived the analytical formula to compute the buffer blocks under any given delay constraint. However, the formula did not count in signal transition which is crucial for high signal integrity. Then, researchers [3] presented a buffer block planning algorithm that meets both the delay and transition time but it only works on a 2-pin net. After that, researchers [4] have improved the idea to cover nets with multiple pins given an existing buffer block plan. The abovementioned techniques provide good buffer repeater results based on a set of user defined parameters such as wire resistance, intrinsic repeater delay, repeater input and output capacitance. However, in recent technology, the parameter can be of many different values. For an example, wire resistance varies depend on the wire layer, width and length, spacing, and repeater driving strengths. Thus, searching for the right parameters itself can be a challenge. Different design with different reliability requirement and operating clock frequency may require different repeater design recipe. It is a tedious task to find an optimum repeater recipe as explained by researcher [5] in his work for 32nm technology. For that reasons, we have introduced a hybrid meta-heuristic technique which uses machine algorithm to automatically learn heuristically and then search for a near optimum recipe.

For interconnect delay that is more than a clock cycle, flop repeater is needed as explained by researchers [6]–[9] in their work which is related to buffer and flop repeater optimization for performance and power consumption. However, they have used only the simplified and ideal models in their analysis. In a real complex SoC design, there are more factors to be considered. For examples, multiple clock domains concern [10], timing margin management [11], and Bit Error Rate (BER) consideration [12]. In an actual SoC product development, to achieve the optimum interconnect design, knowledge on circuit level analysis alone is

insufficient. We have to analyze the product at higher level such as floor-plan and global route congestion as proposed by researchers [13], even up-to architecture and computer aided design (CAD) tool development levels as explained by researchers [14]. However, the abovementioned techniques used are mainly the exact algorithms, which guaranteed to find for every finite size instance of a combinatorial optimization problem an optimal solution in bounded time [15], but are impractical for a complex SoC case today. As we are now at the phase 3 in which a completely new ecosystem emerged during the past decade as reported by International Technology Roadmap for Semiconductors (ITRS) 2015 [16]. With the emergence of fabless companies, the business becoming more competitive as companies have to respond to rapidly changing, complex business requirements, and increased expectation by customers for faster delivery of new and high volume products [16]. Therefore, in the current environment, the development cost and schedule have become key factors to the success of a SoC product.

In this paper, we present a repeater design methodology that is practical for a complex SoC development today. It can produce results with good enough quality, fast turn-around-time, and reduced engineering resources. This is achieved by using simplified design models and hybrid meta-heuristic techniques to minimize the human trial-and-error iterations. The rest of this paper is organized as follows: In Section 2, we explain the challenges of repeater design in a complex SoC in the current business environment. In Section 3, we explain the concept of hybrid meta-heuristic and how it is being used in the proposed new repeater design flow. Section 4 elaborates the hybrid meta-heuristic technique used in searching for the optimum buffer repeater recipe. In Section 5, we introduce the proposed new hybrid meta-heuristic based flop repeater insertion flow and then in Section 6, we will prove the efficiency of the flows on real SoC products. Finally, Section 7 concludes the paper with some directions for future works.

II. REPEATER DESIGN CHALLENGES

The common known challenges of repeater design are:

- Latency – Pipelined interconnects are designed so as to minimize the overall latency of the propagated signals or to satisfy latency constraints given at each driver-receiver pair [8].
- Power – The power consumed by interconnect including repeaters and flip flops gains a growing significance in the total system power [9]. Thus, the power dissipation has become a primary design constraint.
- Signal Reliability – In real circuits, there are many non-ideal behaviors of circuits and signals due to temporal and spatial variations of clock signal (clock skew and jitter), wire delay uncertainty, and variations of timing parameter of sequential elements, which will greatly impact the reliability of wire pipelining scheme [17].
- Area cost – It was projected that over 700k repeaters will be inserted in a single chip for the 70-nm technology. The insertion of that many repeaters will significantly change the floor-plan and placement of a design. Many designs are routing-limited; it may not be feasible to get signals to and out of repeaters due to the limitation in routing resources. Therefore, it is important to consider routing feasibility during the global distribution of repeaters in the floor-planning step [18].

However, in the recent business environment, the following challenges have emerged as primary concerns:

- Design productivity and development cost – As listed in challenges and possible solution Section of ITRS 2.0 2015 [16], 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.
- Design Complexity - As mentioned by researchers [19], in the 2013 ITRS roadmap, the die area of Consumer Portable SoC (SOC-CP) is about 140mm^2 and transistor count is about 2.4B transistors. Recent trends suggest a factor of 1.26$\times$ scaling of logic transistor per core with every process technology node.

In a complex SoC, we may need both buffer and flop repeaters. Buffer repeater insertion can be solely implemented and validated by physical designer, but flop repeater insertion requires additional parties such as micro-architecture designer, floor-planner, timing and clock designer. On top of design, the amount of effort put into communication to converge a SoC cannot be under looked. In this paper, we focus on developing a methodology that can solve the common known challenges listed above with fast turn-around-time and less design as well as communication efforts.

III. HYBRID META-HEURISTIC TECHNIQUES

As described in Section II, repeater design of a complex SoC in the recent business environment is a non-deterministic polynomial-time hardness (NP-Hardness) optimization problem, where no algorithm exists to solve it in polynomial time. The usage of the exact algorithms, such as mathematics 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 [15]. In repeater insertion context, an example of heuristic method is to configure constraints of a layout CAD tool and then rely on its auto place and route (APR) algorithm to build the solution. APR is 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 [20]. Therefore, local search algorithm will simply get caught in the local optima [15] and to solve this problem modern explorative meta-heuristic techniques have been proposed. According to researchers [21], a meta-heuristic is formally defined as an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for

exploring and exploiting the search space. Learning strategies are used to structure information in order to efficiently find near-optimal solutions. There were other researchers attempted to propose a definition as well. Summarizing, researchers [22] outlined fundamental properties which characterize meta-heuristics in the literature of their work and proposed that meta-heuristics are high level strategies for exploring search space by using different methods. In the last few years, the literature reports [20] that a huge number of algorithms have not followed the conventional single meta-heuristic strategy but have combined different algorithmic ideas from both the meta-heuristic as well as outside the meta-heuristic field. Such an algorithm is called as a hybrid meta-heuristic algorithm [23]. Therefore, these algorithms compose exact or heuristic or meta-heuristic strategies with some other meta-heuristic strategies in an attempt to remove each other's weaknesses and merge their strengths.

The proposed repeater design can be divided into two categories that are buffer repeater and flop repeater. In buffer repeater design, we will use Genetic Algorithm (GA), which is a type of population based meta-heuristic algorithms, to search for near-optimal design recipe such as the first repeater size, middle repeater drive strength, repeaters interval distance, metal layers, metal width, spacing, and shielding to produce the near optimum latency with acceptable area, signal integrity, and power tradeoffs. Then, we will deploy heuristic method to statistically estimate the performance of the recipe during actual physical implementation. The buffer repeater hybrid meta-heuristic flow will be elaborated in Section 4. The statistically estimated buffer repeater performance have been used in the floor-planning, global route congestion modeling, micro-architecture fine tuning, and CAD flows development such as placement in APR and flop repeater insertion.

The proposed flop repeater design flow is a hybrid meta-heuristic based flow and will be elaborated in details in Section 5. For high level description, the flow uses meta-heuristic approach to optimize pin placement, heuristic approach to implement channel physical blocks, aka partitions, which are the larger version of buffer blocks [1]–[4] and design the feed-through pins. Based on latency per distance data estimated by the hybrid meta-heuristic flow explained in Section 4, it will use exact algorithm to calculate the pipeline stage count for each feed-through signal in the channel partitions. The output from the flow will eventually be fed-back to micro-architecture designers to be modelled into the design. The key advantages of this flow is that it speeds up the turn-around-time and minimize the engineering efforts, as it allows flop repeater design to be performed by floor-plan and micro-architecture teams, hence skipping the involvement of timing, clock, and physical implementation teams.

IV. BUFFER REPEATER INSERTION METHODOLOGY

Buffer repeater recipe in this paper refers to combination of input parameters such as the first buffer repeater size, buffer repeater drive strength, repeater interval distances, wire metal layers, wire width, spacing, and shielding. Among these parameters, the repeater interval distances are not combinatory. In the prior works [2]–[4], [6]–[8], [11], [18], resistance and capacitance of wires and cells, which are depending on the buffer repeater recipe, have to be predefined by designers for the exact algorithm to find the feasible region for repeater insertion and the optimal interval distances in order to minimize latency and power consumption. However, there are many combination of the repeater input parameters. With a fixed value of these parameters, the exact algorithm can only produce a locally optimized result. In work [5], a heuristic algorithm, which builds Resistance-Inductance-Capacitance (RLC) circuits and then verified using Simulation Program with Integrated Circuit Emphasis (SPICE), is used to find an optimum repeater recipe for 32nm technology. The algorithm is experimented on circuitries for three repeater interval distances only. The optimal buffer recipe is identified through the plotting of Pareto points on the charts of propagation time versus buffer strength. In reality, the repeater interval distance is not a combinatorial value, thus the results of the analysis may not be global optima.

To identify a better buffer repeater recipe, we will need to search in a wider space, in which means more combination of input parameters. Therefore, we have deployed GA, a type of population based meta-heuristic algorithm. GAs are search methods based on principles of natural selection and genetics. GAs encode the decision variables of a search problem into finite-length strings of alphabets of certain cardinality. The strings which are candidate solutions to the search problem are referred to as chromosomes, the alphabets are referred to as genes and the values of genes are called alleles. As described by researchers [24], GAs rely on a population of candidate solutions. The population size, which is usually a user-specified parameter, is one of the important factors affecting the scalability and performance of GAs. To evolve good solutions and to implement natural selection, we need a fitness measure for distinguishing good solutions from bad solutions. Once the problem is encoded in a chromosomal manner and a fitness measure has been chosen, we can start to evolve solutions to the search problem using bio-inspired operators such as reproduction, selection, mutation, and crossover.

In the proposed GA technique, we model a global signal implementation into three parts: driver, trunk, and receiver as shown in Fig. **1**. The recipe for each part is modeled in Chromosomes strings, which include the genes that represent the first buffer, first distance, repeater cell(s) drive strength, interval distance, wire layer, wire width, wire spacing, and shielding. The fittest chromosomes, is the combination of the three sub-chromosomes strings that, without violating the maximum transition (Trans) and Maximum capacitance (Cap) requirements, produces the shortest latency and consumes reasonably low power.

The proposed hybrid meta-heuristic based buffer repeater insertion flow is shown in Fig. 2. Basically, a GA meta-heuristic algorithm is used to search for a near-optimal design

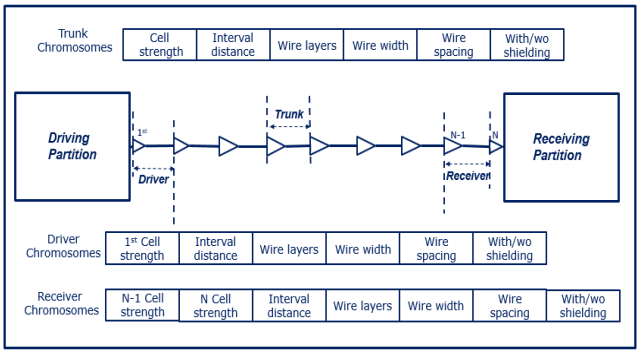

FIGURE 1. A full chip (FC) global signal model in the proposed GA.

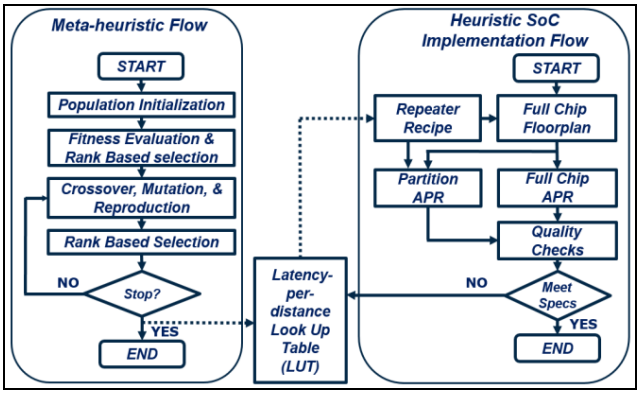

FIGURE 2. The proposed hybrid meta-heuristic based buffer repeater recipe finder flow, which combines meta-heuristic flow with heuristic SoC implementation flow to assist the near optimum recipe searching.

recipe and followed by a heuristic SoC implementation flow to fine tune the recipe. The proposed initial population generation flow is as modeled in Fig. 3. In the flow, we developed scripts to automatically mix and match the genes options to produce chromosomes. For each chromosome, a simple repeater layout, which consist of the buffer repeaters design as described by the chromosome, will be automatically generated using APR CAD tool. After that, we will extract the parasitic resistance-capacitance (RC) of the design and perform static timing analysis (STA).

In the initial reproduction flow, we choose to operate on a 1000um*1000um floorplan, which is an intended trade-off of accuracy for the turn-around-time. The trade-off of accuracy, which is due to non-optimized receiver part, will be rectified using heuristic flow. On top of that, to reduce the size of the initial population, we have preset some of the input values based on know-hows learnt from the previous SoC design experiences. For examples, we combine the first and last stage buffers into the same set and remove the big cells from the set to reduce the risk of placement congestion near to the pins. We set the granularity of the trunk interval distance to 25um to reduce population size and the shielding to always ON to mimic the coupling capacitance effect to improve accuracy in STA.

To simplify the implementation flow development, we set the driver interval distance to half of the trunk interval

Inputs:

First/Last cell strength, *A= {bfn1n06, … , bfn1n32}*
Cell strength, *C= {bfn1n04, … , bfn2n72}*
Interval distance, *I= {100, … , 250} micrometer(um)*
Wire layer pair, *L= {(m4 m5), … , (m10 m11)}*
Wire min Width multiplier, *W= {1, 3}*
Wire min Spacing multiplier, *S= {1, 2, 3}*

Outputs:

Layouts for each Chromosomes string.
Timing reports for each Chromosomes string.

```
Initialization of a layout database
Create design, auto_repeater and set Inputs parameters
Create two pairs of input-and-output ports
Create two nets and connect the two port pairs
Save changes on "auto_repeater"
For each first/last cell strength, a ϵ A
  For each wire width multiplier, w ϵ W
    For each wire spacing multiplier, s ϵ S
      For each wire layer pair, l ϵ L
        For each interval distance, i ϵ I
          Open design "auto_repeater"
          Initialize a 1000um*1000um Floor plan
          Place port pairs at left-right and top-bottom edges
          Insert the first/last cell, a at input/output ports
          Route all the nets with layer pair, l
          Save design as "post_anchor"
          For each cell, c ϵ C
            Open "post_anchor"
            Insert c on nets with first distance i/2 and interval i
            Route all the nets with layer pair constraints, l
            Convert net width to min wire width*multiplier, w
            Shield all nets with min spacing*multiplier, s
            Extract Parasitic RC and Report Timing
            Save design as per the Chromosomes string
            Discard the new changes on "post_anchor"
          End for
          Discard the new changes on "auto_repeater"
        End For
      End For
    End For
  End For
End For
```

FIGURE 3. The initial population generation flow with its inputs simplified based on past experiences to reduce the population size.

distance and the wire used in the driver, trunk and receiver to the same metal layers. With incorporation of these know-hows, the Chromosomes string has been shortened into the format of $\{a\}.\{c\}.\{i\}.\{l\}.\{w\}.\{s\}.\{c_r\}$, where the first/last cell strength $a \epsilon A = \{bfnln06, \ldots, bfnln32\}$; Trunk Cell strength $c \epsilon C = \{bfnln04, \ldots, bfn2n72\}$; Interval distance $i \epsilon I = \{100, 125, \ldots, 250\}$; Wire layer pair $l \epsilon L = \{(m4\ m5), \ldots, (m10\ m11)\}$; Wire min Width multiplier $w \epsilon W = \{1, 3\}$; Wire min Spacing multiplier $s \epsilon S = \{1, 2, 3\}$ and the N-1 cell strength $c_r \epsilon C = \{bfnln04, \ldots, bfn2n72\}$. In the initial population flow, to simplify the implementation, we set $c_r = c$.

In the STA process, based on the location of the input-output port pair, we can identify the direction of the signal paths by either vertical (V) or horizontal (H) and then differential the paths by layer. With a simple script, for each path *I*, we measure its Manhattan distance M_i and latency D_i from

driver-pin to the input of its N-1 cell, skipping the receiver part, then calculate the latency-per-distance $K_i = D_i/M_i$. Upon completion of the STA process, we save the reports into a format which represent their respective Chromosomes string as shown by samples in Table 1. Then, the table will go through a rank-based selection [25] and a fitness evaluation flow. First, we sort the latency in descending order to identify the Chromosomes with the lowest latency. Then, in ascending order, the lowest members will go through a Fitness Function, in which Chromosomes that failed in the maximum transition (Trans) and maximum capacitance (Cap) checks will be filtered out. The first candidate who passes the fitness function will be the overall winner among the initial population and its Chromosomes will be used for Mutation operation in the subsequent optimization.

TABLE 1. Samples of the initial population data.

A	C	I (um)	L	W	S	K (ps/mm)	Trans (ps)	Cap (fF)
bfn1n32	bfn2d64	250	m9	3	1	203.2	0	26.57
bfn1n32	bfn2n64	225	m9	3	1	203.3	0	9.98
bfn1n32	bfn1n48	250	m9	3	1	203.6	0	26.6
...								
bfn1n32	bfn1n48	250	m9	1	1	228.0	0.31	0
...								
bfn1n32	bfn2n48	225	m11	1	1	238.5	0	0
...								
bfn1n32	bfn2d64	225	m10	1	1	246.3	0	0
...								
bfn1n32	bfn2n48	200	m7	1	1	304.0	7.38	0
...								
bfn1n32	bfn1n48	200	m8	1	1	307.4	0	0
...								
bfn1n32	bfn2n48	100	m6	1	1	380.6	0	0
...								
bfn1n06	bfn1d32	100	m5	1	1	673.7	7.11	0
...								
bfn1n06	bfn1n16	100	m4	1	1	786.6	0.88	0
....								
bfl1n32	bfl2n03	250	m4	1	1	1509.3	2417.0	249.1
bfl1n06	bfl2n03	250	m4	1	1	1516.9	2435.5	254.9
bfl1n06	bfl1n03	250	m4	1	1	1520.1	2458.0	255.1

Different SoCs will have different priorities on latency, power, signal reliability and area. For example, in a low power SoCs, power consumption will be relatively higher weightage factor in the rank-based selection. To solve this multi-objective problem, we can use a simple method to aggregate all the criteria into one criteria using a weighted summation [26], that is Cost $S_c = m_k K + m_p P + m_s T + m_a D$, where K is the latency-per-distance; P is the power consumption; T is the worst signal transition; D is the cell area; m_k, m_p, m_s and m_a are user defined weightages as per the SoC priority with all $m > 0$ and $m_k + m_p + m_s + m_a = 1$. In the SoCs used in this paper, we simplified the selection process by setting $m_p = m_s = m_a = 0$. This is because we have considered maximum transition in the proposed fitness function and filtered out any high power consumption and large cells from the search space during the initialization of population. Controlling the solution selection using maximum transition and applying side shielding to signal wires are the key techniques we use to solve signal integrity challenge.

In a complex SoC design that has many global signals, routing tracks could be one of the key factors to die area growth. Thus, using mainly single metal layer for all the global signals is not a practical solution for a cost sensitive SoC. As there are global signals with different clock frequencies, we can optimize die area by using different metal layers as long as its total latency is still within the margin. Therefore, beside the overall winner, we also identify winners for different categories, in which the winners are used for different purposes. First, we categorize the population by wire min width multiplier, $w \epsilon W$. The winner of category $w = 1$ is meant for default data signal routing whereas category $w = 3$ is meant for critical clock signal that will be custom built. The category of min width multiplier$w = 1$, which is meant for data signals, is further categorized by each wire layer $l \epsilon L_m = \{m4, m5, \ldots, m10, m11\}$ and captured into a format as shown by examples in Table 2. This data is meant for floor-planning, similar to the objective discussed by researchers [13], and data signal pipeline designs as elaborated in Section 5.

TABLE 2. Examples of winner by layer and with $w = 1 : w \epsilon \{1, 3\}$.

L_m	Direction	K (ps/mm)	Chromosomes
m4	H	786.6	bfnln06.bfnln16.100.m4.1.1.bfnln16
m5	V	673.7	bfnln06.bfnld32.100.m5.1.1.bfnld32
m6	H	380.6	bfnln32.bfn2n48.100.m6.1.1.bfn2n48
m7	V	304.0	bfnln32.bfn2n48.200.m7.1.1.bfn2n48
m8	H	307.4	bfnln32.bfnln48.200.m8.1.1.bfnln48
m9	V	228.0	bfnln32.bfnln48.250.m9.1.1.bfnln48
m10	H	246.3	bfnln32.bfn2d64.225.m10.1.1.bfn2d64
m11	V	238.5	bfnln32.bfn2n48.225.m11.1.1.bfn2n48

On top of that, the winners for each interval distance $i \epsilon I$ are rank-based selected by wire layer $l \epsilon L$ and captured into a format as shown by examples in Table 3. The table is used for Crossover operation later to find the optimum solution for receiver part of a global signal. In this selection process, we update the fitness function to ensure the trunk cell strength $c \epsilon C$ is be equal or smaller than cell strength c of their respective winner in Table 2.

TABLE 3. Examples of winner by interval distance and with $w = 1 : w \epsilon W$.

L	I (um)	K (ps/mm)	Chromosomes
m11	100	308.3	bfn1n32.bfn1n48.100.m11.1.1.bfn1n48
m11	150	264.1	bfn1n32.bfn1n48.150.m11.1.1.bfn1n48
m11	175	264.3	bfnln32.bfn1n48.175.m11.1.1.bfn1n48
m11	200	255.2	bfnln32.bfn1n48.200.m11.1.1.bfn1n48
m10	100	298.9	bfn1n32.bfn1n48.100.m10.1.1.bfn1n48
m10	150	269.9	bfn1n32.bfn1n64.150.m10.1.1.bfn1n64
m10	175	262.7	bfn1n32.bfn2n48.175.m10.1.1.bfn2n48
m10	200	254.2	bfn1n32.bfn1d64.200.m10.1.1.bfn1d64
m9	100	289.9	bfn1n32.bfn1n48.100.m9.1.1.bfn1n48
m9	150	256.4	bfn1n32.bfn2n48.150.m9.1.1.bfn2n48
m9	175	247.2	bfn1n32.bfn1n48.175.m9.1.1.bfn1n48
m9	200	235.1	bfn1n32.bfn1n48.200.m9.1.1.bfn1n48
m9	225	231.8	bfn1n32.bfn1n48.225.m9.1.1.bfn1n48
m8	100	329.9	bfn1n32.bfn2n48.100.m8.1.1.bfn2n48
m8	150	312.5	bfn1n32.bfn1n48.150.m8.1.1.bfn1n48
m8	175	308.6	bfn1n32.bfn1n48.175.m8.1.1.bfn1n48
m7	100	327.1	bfn1n32.bfn1d32.100.m7.1.1.bfn1d32
m7	150	310.7	bfn1n32.bfn1d32.150.m7.1.1.bfn1d32
m7	175	306.5	bfn1n32.bfn1d32.175.m7.1.1.bfn1d32

To search for the local optimum point, we can deploy the GA Mutation operation on the overall winner's

Chromosomes. In the operation, only the repeater interval distance genes $i \epsilon I$ of the Chromosome will be mutated both incrementally and decreasingly with finer granularity of 5um. Based on the generated offspring's Chromosome, the simple layouts have been automatically generated, extracted, and analyzed through another round of rank-based selection flow and Fitness Function to identify the winner, which is effectively the Pareto point of the latency versus repeater interval distance plot. This Mutation operation can be applied on the winners for different wire layer $l \epsilon\ L$ to optimize the value in Table 2.

We depict that a receiver interval distance i_r, which is the distance between N-1 and N cells as shown in Fig. 1, is depending on the total path distance l_t from driver pin to receiver pin and the trunk interval distance i as $i_r = ((l_t\text{-}0.5i) mod\ i)^* i$.. In a SoC, different global signals can have different total path distance l_t and thus different receiver interval distance i_r. Therefore, we can crossover the N-1 cell strength gene $c_r \epsilon C$ of its chromosomes string with Chromosomes in Table 3 to further optimize the latency of the global signals. In practical, this crossover process shall happen after the global signals have been implemented. By matching the wire layer pair $l \epsilon L$ and approximating the receiver interval distance i_r of each global signal to the trunk interval distance i of the Chromosomes in Table 3, the N-1 cell strength c_r are crossed over with engineering change order (ECO) process.

For a global signal with multiple fan out, we will preprocess the signal by splitting it into multiple single ended signals using script as illustrated in Fig. 4. With that, the Chromosomes of single ended signal found can be re-used on them. For data signal that has very high fan-out, the script will build a multi stages tree but this may increase the latency significantly. Thus, for the very high fan-out signal, which is usually multi-cycle path that does not need to be optimized for latency, we implement it using other technique, which will not be covered in this paper.

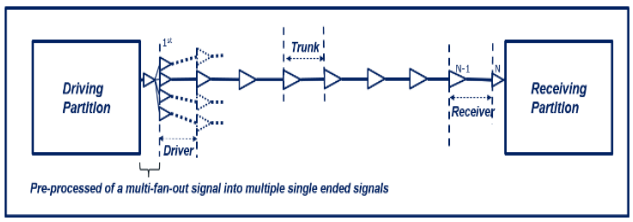

FIGURE 4. The conversion process of a multiple-fan-out signal into multiple single ended signals to reuse the recipe found by the new meta-heuristic technique.

The look up tables (LUTs) in Fig. 2 are essentially Table 2 and Table 3 which we can use to generate the default repeater recipes for a design using a heuristic SoC implementation flow. The repeater recipes are referred to the configuration of the CAD APR tool for buffer insertion and reference for custom circuit design for critical clock signals in this paper. With the heuristic flow, we can rectify the latency per distance K (ps/mm) value of Table 2 to cover the inaccuracy of receiver modeling abovementioned and any routing overhead added by the CAD tool. One of the key overheads is the usage of lower metal to solve routing congestion. An example of the heuristic results is shown Fig. 5, where 18 thousands data points are extracted from one of the channel partitions in a 10nm SoC. The global signals in the channel partition have been implemented using Chromosomes with $L = m4$in Table 2. From the data, we depict that the minimum latency per distance K (ps/mm) at 1000um distance is correlating with the 786.6ps/mm value in Table 2. However, for actual implementation, we should use the average value of the latency per distance K instead. From the heuristic data, after filtering out the outliers caused by routing congestions, the latency per distance K (ps/mm) for Chromosomes with $L = m4$ is rectified to 1000ps/mm.

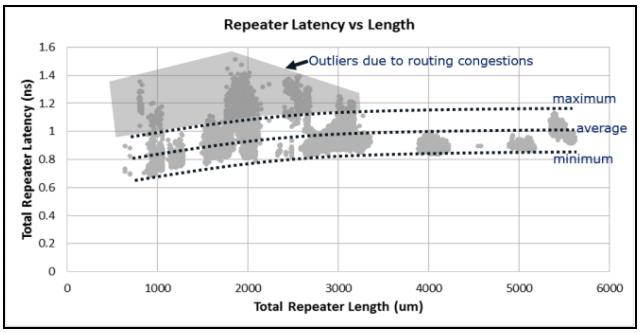

FIGURE 5. An example of heuristic data extracted from 18 thousands buffer repeater paths implemented using Chromosomes with layer $L = m4$. It is used to rectify the latency-per-distance K (ps/mm) value in Table 2.

V. FLOP REPEATER INSERTION METHODOLOGY

An overview of the proposed new hybrid meta-heuristic based flop repeater design flow is shown in Fig. 6. The flow is meant for a complex SoC that has many global signals to be flop repeated, up to the situation where manual handling is no longer practical.

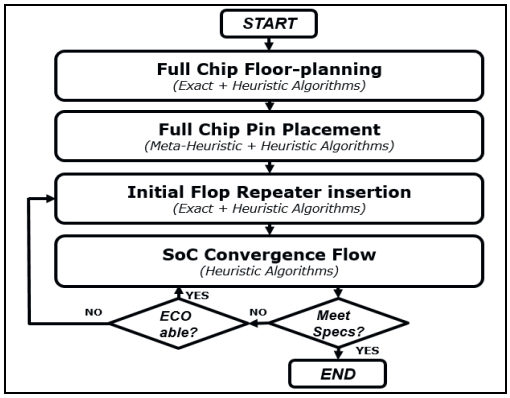

FIGURE 6. The overview of the proposed hybrid meta-heuristic based flop repeater insertion flow, which is built for a complex SoC.

First, we will read LUTs, as shown in Table 2, into full chip (FC) floor-planning process. In the process, for each

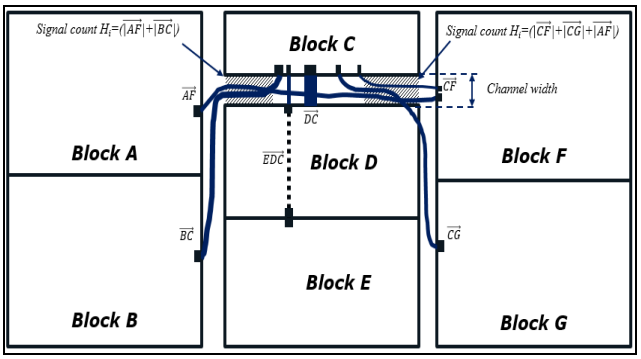

FIGURE 7. An example of full chip floor-plan with global signals labeled to ease the explanation of signal latency and channel width calculation.

block-to-block path group *i,* for example in Fig. 7:

$$i = \left\{\overrightarrow{BC}, \overrightarrow{AF}, \overrightarrow{EDC}, \overrightarrow{CG}, ...\right\}$$

We estimate the average horizontal h_i and vertical v_i distances of its Manhattan route and then calculate the latency $D_i = K_h h_i + K_v v_i$ based on the latency-per-distance values $K_h = \{K_{m4}, K_{m6}, K_{m8}, K_{m10}\}$ and $K_v = \{K_{m5}, K_{m7}, K_{m9}, K_{m11}\}$ in Table 2. With that, we can fine tune the relative placement of the blocks to meet their respective timing requirements. In this paper, we use only the exact and heuristic algorithms. As our meta-heuristic technique based FC floor-plan flow, which can be an enhancement to work [27], is still work in progress. After that, we will calculate the width of each channel $W_i = K_p H_i$, where K_p is a constant of routing-track-per-um for the particular process node after minus off the tracks allocated for power straps whereas H_i is the signal count of the highest routing density point on the channel. For the example in Fig. 7,

$$Hi = MAX\left(\left(\left|\overrightarrow{AF}\right| + \left|\overrightarrow{BC}\right|\right), \quad \left(\left|\overrightarrow{CF}\right| + \left|\overrightarrow{CC}\right| + \left|\overrightarrow{AF}\right|\right)\right).$$

Based on the value of W_i, we can adjust the channel width accordingly. Another alternative way to optimize for area is by altering the shape of the blocks to produce rectilinear channels but at the cost of engineering effort.

After the floor-plan initialization with either virtual flat, black-boxed or reduced net-list, by using CAD tool, we heuristically perform global route, push down of signals into partitions as feed-through, and creation of the signal pins at partitions edge. We have developed an automation script to extract the initial pin locations and modeled them into LUTs. With the LUTs, we can maintain similar pin locations across different design versions and even for different design input format such as flattened, reduced, or black-boxed net-list. On top of that, we optimize the pin placement LUTs by using mean shift clustering which is also a meta-heuristic algorithm. Mean shift is a simple iterative procedure that shifts each data point to the average of data points in its neighborhood as described by researchers [28]. We use mean shift algorithm to reduce efforts in the visual inspection of the pin clustering results and manual editing of pin constraints.

The proposed mean shift algorithm based new meta-heuristic FC pin optimization flow is shown in Fig. 8. Basically, the flow optimizes pins location through iterations of constraints generation, fast heuristic placement, quality checks and then rank-based selection [25] against the previous best-known constraints. From the ranking, the better constraints will be logged and used for pin placement in the next iteration. In the case if the new constraints failed to produce a better result, the flow will restore the previous best-known constraints and run a mathematic calculation based constraints optimization script to clear if any potential deadlock. The script manages to filter out outlier data points such as wrong edge pins and align the pin constraints between blocks that facing each other directly. Then, the flow will proceed with the new pin constraints and iterates until user defined loop count is met.

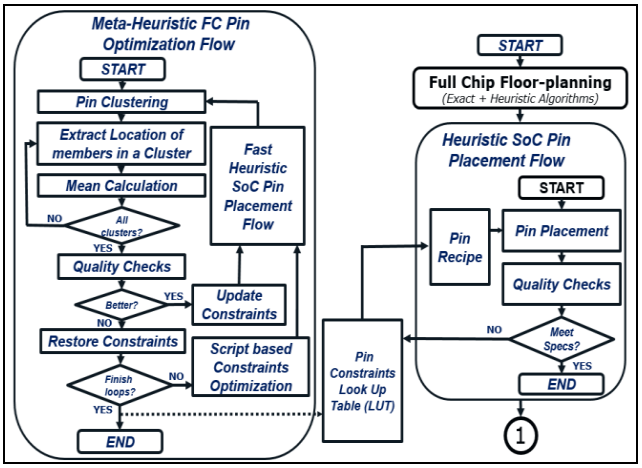

FIGURE 8. The proposed mean shift meta-heuristic technique based pin optimization and heuristic SoC pin placement flows.

Fig. 9 shows a model of two pin clusters, going through three iterations of simple mean shift flow and eventually converged into a better distribution. In this flow, input sets are the pin clusters, which are clustered based on the timing path and bus name on single axis. The data points of pin clusters are finite numbers. Therefore, instead of finding the highest density point, we calculate the simple mean of every

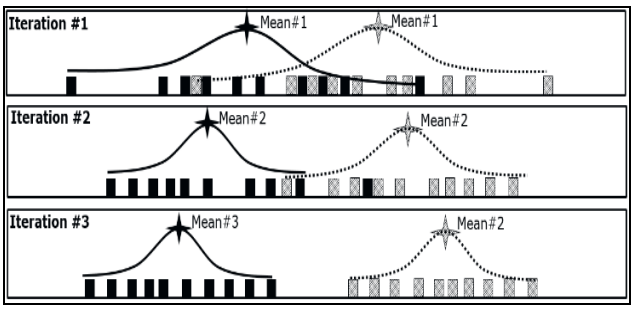

FIGURE 9. A mean shift model for two signal pin clusters where overlaps are resolved as the flow iterates.

pin clusters at once. Then, based on the new mean values, we will regenerate the new pin constraints which we will use to heuristically regenerate new pin placements. Through multiple iterations of the mean shifting and constraints regeneration cycle, we can converge the pin placement and the final pin placement will be extracted into pin constraint LUTs which will be used in the actual heuristic SoC pin placement flow.

The pin clusters are labeled with a specific naming convention, which combines the signal driver partition name, the receiver partition name, and the bus name together. In the proposed pin constraint LUTs, each pin cluster will be modeled with two variables, which represent their respective partition edge and offset range. For the pin cluster example illustrated in Fig. 10, the variables will be:

set *<driving_partition>_2_<receiving_partition>_<bus_name>_edge 4.0*

set *<driving_partition>_2_<receiving_partition>_<bus_name>_offset {Xmin, Xmax}*

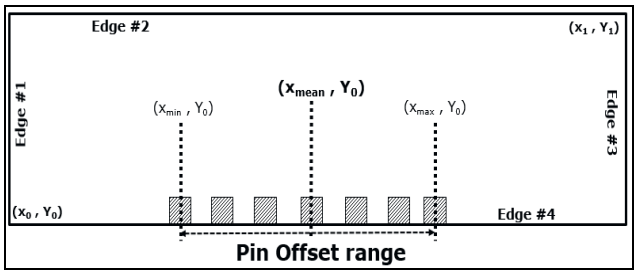

FIGURE 10. The model of a pin cluster constraint in the proposed mean shifts technique.

With $x_{min} = x_{mean}\text{-}x_0\text{-}[(n/2)/K_p]$ *and* $x_{max} = x_{mean}\text{-}x_0\text{+}[(n/2)/K_p]$ where n is the total number of pins in the cluster, X_{mean} is the simple Mean value of the axis-coordinate whereby in this example it is the x-coordinate of all the pins in the cluster, and K_p is a constant of routing-track-per-um that is process node dependence.

In the proposed pin quality checks step shown in Fig. 8, we are using a formula derived from the standard deviation of a pin frequency histogram. For the pin distribution of a cluster as illustrated in Fig. 11, we can bin the pins for every 5 um from the x_{mean} and plot a histogram of pin-number-per-bin versus pin distribution distance as in Fig. 12.

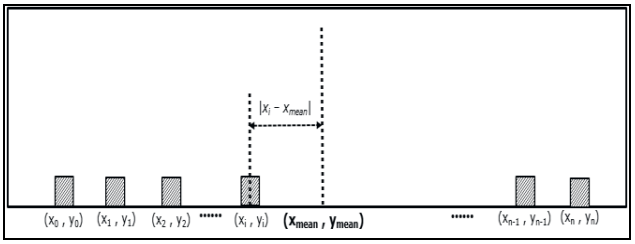

FIGURE 11. The model used to explain the formula used in the proposed simplified density quality checking step.

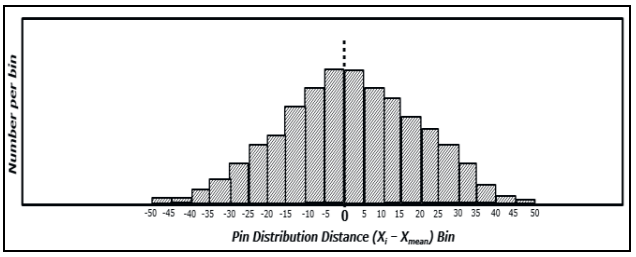

FIGURE 12. An example of histogram of number-per-bin versus pin distribution distance bin. (This is for illustration only)

From the histogram, we can calculate the standard deviation σ as in (1). If a distribution is more spread out, then the deviation will be bigger.

$$\text{standard deviation}, \sigma = \sqrt{\frac{\sum_{i=1}^{n} (X_i - Xmean)^2}{n}} \tag{1}$$

However, other than the spreading of pin, the pin cluster with larger pin counts will have larger standard deviation σ value than smaller cluster even both of them have an optimized pin placement. Therefore, to assess the quality of a pin placement, we will calculate the average distance (um) per pin. Based on normal distribution 68-95-99.7 three sigma rule [29], probability $P(x_{mean}\text{-}3\sigma \leq x \leq x_{mean} + 3\sigma) \approx 0.9973$. Thus, we can calculate the average distance-per-pin for cluster j with (2):

$$\text{Average Distance per pin}, Aj = \frac{3\sigma * 2}{n} = \frac{6\sigma}{n} \tag{2}$$

In the proposed pin quality checks as in (3), any cluster with distance-per-pin A_j more than double of the $1/K_p$, where K_p is a constant of routing-track-per-um, is considered non-optimized and will be logged into a file for deadlock debugging. Most of the deadlocks are due to large congestion hot spot. To clear the deadlocks, we have to pause the cycle and manually edited some of the LUTs values before continue.

$$\text{Pin Quality} = \begin{cases} Good; & Aj \leq \dfrac{2}{Kp} \\ Bad; & Aj > \dfrac{2}{Kp} \end{cases} \tag{3}$$

As shown in the meta-heuristic flow in Fig. 8, to decide whether to keep or discard the current constraints, we will calculate the total distance-per-pin A_{total} value of all the clusters as in (4) and rank A_{total} against the Golden Total distance-per-pin A_{golden} of the previous best-known constraints as in (5). The current constraints is considered better if A_{total} is smaller than A_{golden}. In that case, A_{total} value will be logged as the new A_{golden} and same as the LUTs.

$$\text{Total Distance per pin}, Atotal = \sum_{j=1}^{n} Aj \tag{4}$$

$$\text{Golden LUT} = \begin{cases} \text{Current LUT}; & Atotal \leq Agolden \\ \text{Golden LUT}; & Atotal > Agolden \end{cases} \tag{5}$$

After the flow finishes as per user defined loop counts, the best-known LUTs logged is ready for deployment in the actual FC pin placement. The LUT is reusable across different register transfer level (RTL) versions. We depict that the proposed mean shift based pin optimization flow focuses on optimization of bus pins. For the remaining pins, such as single bit, source synchronized, high-fan-out, tied-high/low, and floating pin, their placement constraints are handled using conventional global route based SoC design planning methodology.

After pin placement optimization as shown in Fig. 6, we will proceed with initial flop repeater, aka pipeline, insertion flow. With a reasonably good bus pin grouping, we can reduce the logic error rate in the pipeline insertion and consequently minimize the engineering efforts in analysis and filtering of the pipeline insertion results. As an overview, the initial pipeline insertion flow, as shown in Fig. 13, takes in three different LUTs, which will be explained later, as references to process the current FC floor-plan and eventually produce a LUT which contains the information needed for pipeline RTL modeling.

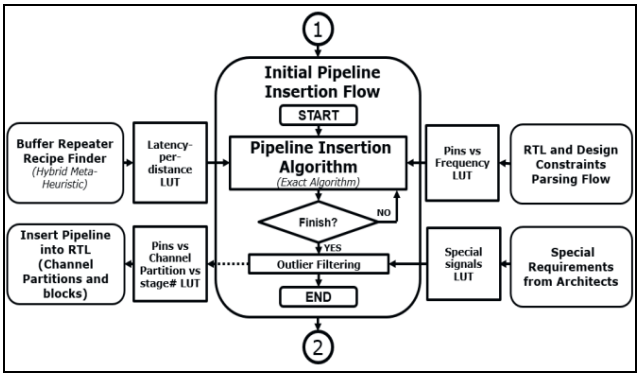

FIGURE 13. The initial pipeline insertion flow which automatically identifies the partition to insert flop repeaters and the flop stages.

Prior to the pipeline insertion step, the FC floor-plan should have channel partitions inserted, as shown in Fig. 14(b), with feed-through signal pushed down and pin properly grouped based on the pin constraints LUT.

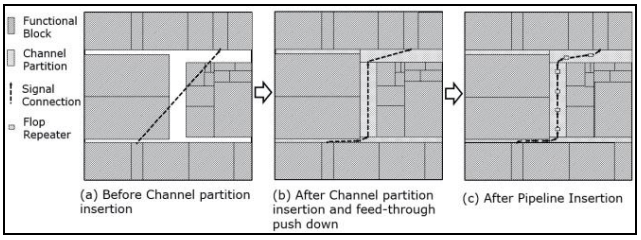

FIGURE 14. A model which illustrates the main pipeline insertion steps in the initial pipeline insertion flow.

To enable a more accurate pipeline stage calculation, we need a pins-vs-frequency LUT from register-transfer level (RTL) designer. Each line in the pins-vs-frequency LUT is following the format of *<path type={sync,async}>, <Bus name>, <Bit Count>, <Driving Pin Name>, <Receiving Pin Name>, <Driving Clock domain>, <Receiving Clock Domain>, <Driving Clock Freq.>, <Receiving Clock Freq.>,* which models the timing and connectivity of a top level signal. After gathering of a complete pins-vs-frequency LUT, we use the algorithm in Fig. 15 to automatically design the pipeline.

Inputs:

FC Floor-plan with signal pushed down into partitions. *See Fig. 14(b).*

$K_l \in$ latency-per-distance LUT with $l \in L = \{m4, m5, \ldots, m11\}$.

Pins-vs-Frequency LUT

Constants:

K_{scale}>1.0 is the scaling factor for slow corner and aging effect.
K_{route}<1.0 is the scaling factor for statistical routing overhead.
T_{skew} is the budgeted systematic clock skew
T_{jitter} is the budgeted clock jitter
T_{misc} is the budgeted miscellaneous uncertainty
Set Total timing uncertainty $T_{uncertainty} = T_{skew} + T_{jitter} + T_{misc}$

Outputs:

Pipeline stages LUT.

Open FC Floor-plan Database
For each line *i* of Pins-vs-Frequency LUT
  If *path type* = "*sync*" **Then**
    Set Clock Period $T_{clk} = 1/(<clock\ Freq> * K_{scale})$
    Set timing margin-per-stage $T_{margin} = (T_{clk} - T_{uncertainty}) * K_{route}$
    Trace signal to identify partition list B_i and respective pin list P_i that it has fed-through
    Set feed-through string S_i=""
    For each partition b_i with $b_i \in B_i$
      Open b_i database
      Calculate Manhattan distance M_{ib} between Pins $p_i : p_i \in P_i$
      Set total delay $D_{bi} = M_{bi} * MAX(K_l) = M_{bi} * K_{m4}$
      Set pipeline stage for the partition $N_{bi} = D_{bi}/T_{margin}$
      Set $S_i = S_i$."$<b_i, N_{bi}>$"
      Close b_i database
    End For
    Write line *i* into Pipeline Stage LUT with S_i appended
  Else
    Calculate Manhattan distance from *Driver* to *Receiver Pins*, M_i
    Set total delay $D_i = M_i * MAX(K_l) = M_i * K_{m4}$
    Write line *i* into Pipeline Stage LUT with D_i appended
  End If
End For
Close FC Floor-plan Database

FIGURE 15. The algorithm used in the proposed initial flop repeater insertion flow.

First, for each line in the LUT, the algorithm traces the connectivity to identify physical blocks that the signal will be feeding through, including channel and functional blocks. Then, for each partition that the signal has fed-through, it will calculate the Manhattan distances and followed by the total signal latency with referring to the latency-per-distance LUT as in Table 2, which is generated by the buffer repeater insertion flow in Section 4. After that, depending on the *<path type={sync,async}>*info, the algorithm will proceed with the pipeline stages calculation if the signal is a *sync* (synchronized) path or skip if it is an *async* (not synchronized) path. During pipeline stages calculation, the algorithm will calculate the timing margin for each pipeline stage. With that, the algorithm can calculate the minimum pipeline stages needed for the signal in

each partition it has fed-through. After that, the flow will write out the result into a pipeline stages LUT with the format of *<path type={sync,async}>, <Bus name>, <Bit Count>, <Driving Pin Name>, <Receiving Pin Name>, <Driving Clock domain>, <Receiving Clock Domain>, <Driving Clock Freq.>, <Receiving Clock Freq.>* $<b_1N_{b1}><b_2N_{b2}>\ldots<b_iN_{bi}>$, which is basically the combinational of input line i from Pins-vs-Frequency LUT with S_i string that represents the feeds-through sequence of a signal and the number of pipeline stages N_{bi} in each partition $b_i \epsilon B_i$ that the signal has fed-through.

Upon completion of all the signal listed in the pins-vs-frequency LUT, we will use simple scripts to parse and edit the pipeline stages LUT to ensure signals that belong to the same *<Bus name>* are having the same pipeline stage count and feed-through path. For sanity check, *<Bit Count>*is used to cross check against the number of lines in the output pipeline stage LUT for *<Bus name>*. Besides, we also check the output LUT against the special signal LUT which is provided by architect to highlight the critical signals that have stringent pipeline stages limitation due to system performance requirement. The proposed script will automatically scale down the number of pipeline stages of the critical signals to meet their respective design requirements and at the same time write out special metal layer constraints for those signals to be loaded into APR for correct implementation. Any critical signals that are not able to meet architect's special requirements will be flagged and logged into error report file for debugging. Finally, the pipeline stages LUT will be consumed by the flow as shown in Fig. 16, in which the LUT will be used to model the pipeline stages in RTL, synthesized into gate level netlist, and then physically implemented using the conventional Heuristic SoC Convergence flow.

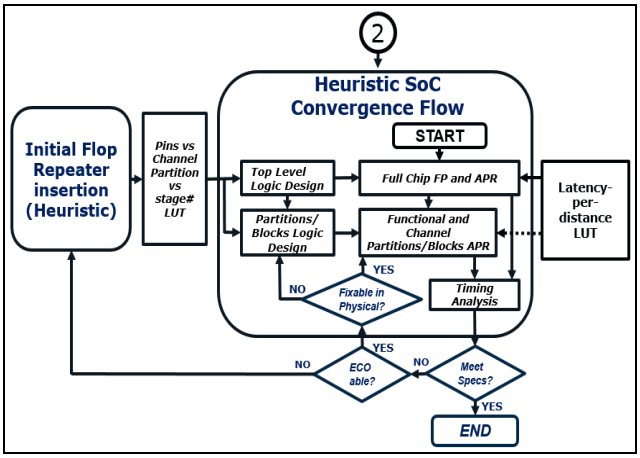

FIGURE 16. A heuristic SoC convergence flow which takes in the pipeline stages LUT to model the pipeline in RTL and then the latency-per-distance LUT for the physical implementation of the repeaters.

VI. RESULTS

To prove the effectiveness of the proposed new hybrid meta-heuristic technique based buffer repeater insertion flow, we carried out an experiment on a taped-out 14nm SoC which has been implemented with only the conventional heuristic SoC flow. We have performed the proposed GA technique to identify the Winner Chromosomes and prepared LUTs for the 14nm nodes. Based on the LUTs, we have revamped the buffer repeater insertion of its global signals using the same conventional heuristic SoC flow and then compared the new latency results with the original one. The path-to-path latency comparison data is shown in Fig. 17, where we depict that the proposed new technique has managed to further improve 43% of the total paths.

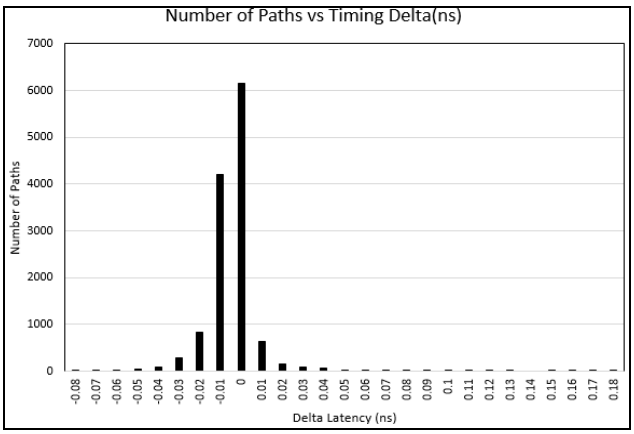

FIGURE 17. The path-to-path latency comparison between the proposed hybrid meta-heuristic technique vs a conventional heuristic technique in a 14nm SoC.

Fig. 18 shows an example of progressive result from the proposed Mean shifts Clustering Based Pin Optimization Flow on a 14nm SoC whereas Fig. 19 show a few examples of the final pin placement results.

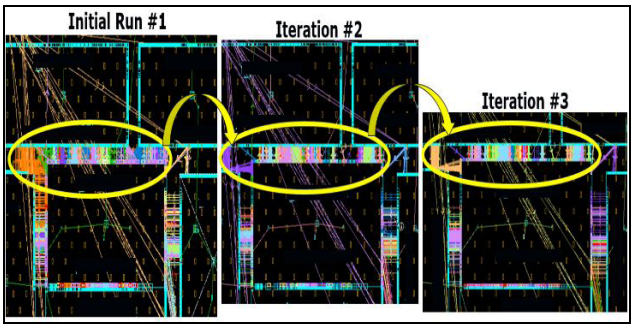

FIGURE 18. A progressive result in the proposed mean shift clustering based pin optimization flow. The spreading of busses are reducing with the iterations of the flow.

Table 4 shows the effectiveness of the flow in stabilizing the pin location across two RTL design versions, whereby less than 9% of pins have moved beyond 250um and most of the pins that moved more than 450um are intended.

We have deployed the hybrid meta-heuristic based flop repeater insertion flow in Fig. 6 on a very complex 10nm SoC which has more than 100k global signals that require flop repeaters. Table 5 shows the results of the flop and buffer repeater insertion after the default iteration of APR flow before involving engineer in debugging and optimiza-

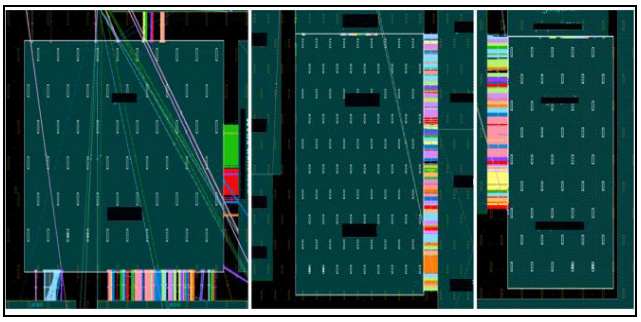

FIGURE 19. A few examples of the final results from the proposed mean shift clustering based pin optimization flow. Busses are grouped together and color coded differently.

TABLE 4. The comparison of pin placement for two RTL versions.

Block	<50um	50-149um	150-240um	250-349um	350-449um	>450um
A	375	491	18	61	1	209
B	0	401	5	0	0	10
C	1	887	7	6	0	103
D	412	1	0	0	0	0
E	13	0	0	0	0	3
F	14	1111	605	1	3	252
G	406	0	0	0	0	10
H	361	1021	329	17	0	115
I	447	0	0	0	0	0
J	294	0	350	0	0	21
K	524	0	1063	16	0	130
L	809	37	244	0	0	308
M	638	159	77	20	0	1
N	369	0	2512	28	0	82
O	0	1599	93	31	0	68
Total	4663	5707	5303	180	4	1312
%	27.15%	33.24%	30.88%	1.05%	0.02%	7.64%

TABLE 5. The results of the flop repeater insertion on a 10nm SoC.

Block	Flops Repeater	Buffer + Flops Repeater	Ports	WNS Setup (ps)	Violated Path	TNS Setup (ps)
A	33408	179608	22019	16.2	0	0
B	16945	122172	18549	95.4	0	0
C	52397	335204	13673	-381.7	22	-517.6
F	28004	153571	31906	155.0	0	0
G	3380	42198	1987	-2.1	2	-3.7
H	13483	180385	9566	-442.1	2986	-181680
I	2010	21349	1988	-3.1	4	-7.6
M	26834	330222	39910	-148.7	405	-12933
N	3799	65980	11122	-12.7	1	-12.7
O	146221	941935	60521	-9.5	6	-26.29
Q	212593	1356807	62321	-8.92	5	-19.98
Total	539074	3729431	273562		3431	-195201

tion work. From this case study, we depicted that the flow correctly inserted over 500 thousands flop repeaters but less than 0.6% of the total paths need further optimization and the worst negative slack (WNS) and total negative slack (TNS) are fixable with just physical design optimization. Besides, the whole flop repeater insertion process including flow development is completed by a team of five engineers including two RTL and three physical designers within two months, saving at least 30 man months of efforts compare to the previous project.

VII. CONCLUSION AND FUTURE WORK

In this paper, we present a new buffer and flop repeater insertion methodology for a complex SoC with a hybrid meta-heuristic technique. We used meta-heuristic algorithms on simplified design models to quickly find a near optimum recipe, then use heuristic algorithms on realistic design samples to rectify the recipe by statistically model in the unobvious overheads, and finally exact plus heuristic algorithms to filter outliers and accurately apply the recipe into the actual designs. With this technique, we manage to insert repeaters into two SoCs in 14nm and 10nm technology nodes with a "good enough" quality at low engineering cost and turn-around-time.

Beside repeater insertion, we are pursuing extensions of the hybrid meta-heuristic technique into different areas in a complex SoC physical design, especially in areas which require extensive engineer effort and time in searching for a converged solution, such as multi-hierarchical hard macro placement, full chip clock distribution design, and multi hierarchical floor-plan optimization for timing convergence.

ACKNOWLEDGMENT

E. Keong would like to thank Joe Murphy, Subeer K. Patel, T. W. Tan, Robert Cone, Kevin Chao, Jayson Strayer, James Lim, Giang T. Nguyen, Kevin P. Svilich, Kah Weng Ng, Yen Chang Goh, Jugantor Chetia, Jack Hong, Yi Hong, Peng Cheng Song, Alice Chiok, 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] F. F. Dragan, A. B. Kahng, I. Mandoiu, S. Muddu, and A. Zelikovsky, "Provably good global buffering using an available buffer block plan," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design (ICCAD)*, Nov. 2000, pp. 104–109.

[2] J. Cong, T. Kong, and Z. D. Pan, "Buffer block planning for interconnect planning and prediction," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 9, no. 6, pp. 929–937, Dec. 2001.

[3] P. Sarkar and C. Koh, "Repeater block planning under simultaneous delay and transition time constraints," in *Proc. Design, Automat. Test Eur. Conf. Exhib.*, 2001, pp. 540–544.

[4] F. F. Dragan, A. B. Kahng, I. Mandoiu, S. Muddu, and A. Zelikovsky, "Provably good global buffering by generalized multiterminal multicommodity flow approximation," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 21, no. 3, pp. 263–274, Mar. 2002.

[5] D. Deschacht, "Optimum repeater insertion to minimize the propagation delay into 32 nm RLC interconnect," in *Proc. IEEE Elect. Design Adv. Packag. Syst. Symp. (EDAPS)*, Dec. 2011, pp. 1–4.

[6] R. Lu, G. Zhong, C. Koh, and K. Chao, "Flip-flop and repeater insertion for early interconnect planning," in *Proc. Design, Automat. Test Eur. Conf. Exhib.*, 2002, pp. 690–695.

[7] P. Cocchini, "Concurrent flip-flop and repeater insertion for high performance integrated circuits," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Nov. 2002, pp. 268–273.

[8] P. Cocchini, "A methodology for optimal repeater insertion in pipelined interconnects," *IEEE Trans. Comput.-Aided Design Integr.*, vol. 22, no. 12, pp. 1613–1624, Dec. 2003.

[9] W. Liao and L. He, "Full-chip interconnect power estimation and simulation considering concurrent repeater and flip-flop insertion," in *Proc. Int. Conf. Comput.-Aided Design (ICCAD)*, 2003, pp. 574–580.

[10] S. Hassoun, C. J. Alpert, and M. Thiagarajan, "Optimal buffered routing path constructions for single and multiple clock domain systems," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2002, pp. 247–253.

[11] M. C. Hon, "Spec-based flip-flop and latch repeater planning," in *Proc. Asia South Pacific Conf. Design Automat.*, Jan. 2006, p. 6.

[12] J. Xu, A. Roy, and M. H. Chowdhury, "Interactive presentation: Analysis of power consumption and BER of flip-flop based interconnect pipelining," in *Proc. Design, Automat. Test Eur. Conf. Exhib.*, 2007, pp. 1–6.
[13] R. Lu and C.-K. Koh, "Interconnect planning with local area constrained retiming [logic IC layout]," in *Proc. Design, Automat. Test Eur. Conf. Exhib.*, Mar. 2003, pp. 442–447.
[14] J. Xu and M. H. Chowdhury, "Latch based interconnect pipelining for high speed integrated circuits," in *Proc. IEEE Int. Conf. Electro/Inf. Technol.*, May 2006, pp. 295–300.
[15] G. I. Zobolas, C. D. Tarantilis, and G. Ioannou, "Exact, heuristic and meta-heuristic algorithms for solving shop scheduling problems," in *Meta-Heuristics for Scheduling in Industrial and Manufacturing Applications* (Studies in Computational Intelligence), vol. 128. Berlin, Germany: Springer, 2008, pp. 1–40.
[16] (2015). *The International Technology Roadmap for Semiconductors 2.0: Executive Report*. [Online]. Available: http://www.itrs2.net/
[17] J. Xu, A. Roy, and M. H. Chowdhury, "Optimization technique for flip-flop inserted global interconnect," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2008, pp. 3386–3389.
[18] P. Sarkar and C.-K. Koh, "Routability-driven repeater block planning for interconnect-centric floorplanning," *IEEE Trans. Comput.-Aided Design Integr.*, vol. 20, no. 5, pp. 660–671, May 2001.
[19] 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.
[20] 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.
[21] I. H. Osman and G. Laporte, "Metaheuristics: A bibliography," *Ann. Oper. Res.*, vol. 63, pp. 511–623, Oct. 1996.
[22] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Comput. Surv.*, vol. 35, no. 3, pp. 268–308, 2003.
[23] G. R. Raidl, "A unified view on hybrid metaheuristics," in *Hybrid Metaheuristics* (Lecture Notes in Computer Science), vol. 4030. Gran Canaria, Spain: Springer, 2006, ch. 1, pp. 1–12.
[24] K. Sastry, D. E. Goldberg, and G. Kendall, "Genetic algorithms," in *Search Methodologies*, 2nd ed. London, U.K.: Springer, 2014, ch. 4, pp. 93–117.
[25] A. Petrowski and S. B. Hamida, "Evolutionary algorithms," in *Metaheuristics*. Cham, Switzerland: Springer, 2016, ch. 6, pp. 115–178.
[26] A. Petrowski, "Extensions of Evolutionary Algorithms to Multimodal and Multiobjective Optimization," in *Metaheuristics*. Cham, Switzerland: Springer, 2016, ch. 11, p. 314.
[27] G. Leary, K. Srinivasan, K. Mehta, and K. S. Chatha, "Design of network-on-chip architectures with a genetic algorithm-based technique," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 17, no. 5, pp. 674–687, May 2009.
[28] Y. Cheng, "Mean shift, mode seeking, and clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 17, no. 8, pp. 790–799, Aug. 1995.
[29] F. P. Miller, A. F. Vandome, and M. John, *68-95-99.7 Rule*. VDM Publishing, 2010.

ENG KEONG TEH was born in Penang, Malaysia, in 1976. He received the B.Eng. degree (Hons.) in electrical and electronics engineering from Universiti Teknologi Malaysia, Skudai, Malaysia, in 1999, and the M.Sc. degree in micro-electronics engineering from Multimedia University, Cyberjaya, Malaysia, in 2010. He is currently pursuing the Ph.D. degree in electrical and electronics engineering with Universiti Sains Malaysia, Penang. He has 19 years industrial experiences. He is currently a Senior Staff Design Engineer for Intel Microelectronic Malaysia Sdn. Bhd., Penang. His current research interest is 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 within the School of Electrical and Electronic Engineering, Universiti Sains Malaysia. His research interests are in the field of micro/nanoelectronics fabrication, IC design, modeling, and simulation of semiconductor devices and sub-Thz & Thz MMIC oscillator based on resonant tunneling diodes.

MOHAMED FAUZI PACKEER MOHAMED (M'16) was born in Kuala Lumpur, Malaysia, in 1978. He received the B.Eng. degree (Hons.) in electrical and electronics engineering from the Universiti Tenaga Nasional Kajang, Selangor, Malaysia, in 2002, the M.Sc. degree in electronics system design engineering from Universiti Sains Malaysia, Penang, in 2010, and the Ph.D. degree in electrical and electronics engineering from The University of Manchester, Manchester, U.K., in 2015. In 2015, he joined the School of Electrical and Electronics Engineering, Universiti Sains Malaysia, 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) at the School of Electrical and Electronic Engineering, USM. His research interests include intelligent systems, image processing, neural network, biomedical engineering, intelligent diagnostic systems, and algorithms. As of now, he has led his Imaging and Intelligent System Research Team Research Group to publish at both national and international arena. He has published over 130 manuscripts indexed in ISI out of a total of more than 250 publications. 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 invited speaker in four international conferences, appointed as a Ph.D. external examiners, appointed as an editorial board for four international journals, and appointed as a reviewers for over 80 ISI indexed journals. He was also appointed as a mentor for two research grants from Malaysian local universities. As a Research Supervisor, he has successfully graduated 17 Ph.D. and 20 master's (by research mode) students.