Digital Compatible Synthesis, Placement and Implementation of Mixed-Signal Time-Domain Computing

Zhengyu Chen, Hai Zhou, and Jie Gu
Department of Electrical and Computer Engineering, Northwestern University, Evanston, IL, USA
zhengyuchen2015@u.northwestern.edu, {haizhou, jgu}@northwestern.edu

ABSTRACT

Mixed-signal time-domain computing (TC) has recently drawn significant attention due to its high efficiency in applications such as machine learning accelerators. However, due to the nature of analog and mixed-signal design, there is a lack of a systematic flow of synthesis and place & route for time-domain circuits. This paper proposed a comprehensive design flow for TC. In the front-end, a variation-aware digital compatible synthesis flow is proposed. In the back-end, a placement technique using graph-based optimization engine is proposed to deal with the especially stringent matching requirement in TC. Simulation results show significant improvement over the prior analog placement methods. A 55nm test chip is used to demonstrate that the proposed design flow can meet the stringent timing matching target for TC with significant performance boost over conventional digital design.

1 INTRODUCTION

Traditional digital circuits rely on the scaling of technology and supply voltage *Vdd* to improve the power consumption of the circuits, following the energy consumption equation of $\alpha CVdd^2$ where C represents the capacitance of the circuit and α is the associated activity factor. As the technology scaling slows down, the energy consumption for digital circuits has reached a bottleneck leading to the urgent need for alternative computing methods. For example, approximate computing provides a good tradeoff between power consumption and accuracy [1]. However, such a technique still follows conventional Boolean operation principles and does not fundamentally change the energy limitation for digital circuits.

Analog computing, which encodes information in analog voltage, provides another solution for energy efficient computing. Numerous examples have shown that analog computing can exceed the energy efficiency of digital design [2]. Unfortunately, analog computing suffers from the issues such as static power consumption, and incompatibility to automatic digital design flow.

Recently, a new class of computing, mixed-signal time-domain computing (TC) emerges as a promising alternative to the existing computing methods [3-6]. TC utilizes digital circuits to encode and process data in time domain. Essentially, TC is similar to analog computing as the data is linearly encoded in a signal line rather than multi-bit binary signals. Benefit from the usage of digital circuits, TC offers the digital compatibility and the technology scalability.

Despite of many existing demonstration of highly efficient operation using TC [3-6], most of existing work for time-domain computing is based on analog/mixed-signal design flow, which requires significant manual design and layout effort. This is partially due to the stringent timing control requirement of the technology leading to the difficulty of adoption into a large-scale design. Hence, it is important to develop a comprehensive design methodology for the automatic synthesis, place & route for TC. It is worth to mention that time-based design has been well explored in traditional mixed-signal circuits such as all digital phase-locked loops (ADPLL), ring-based analog-digital converter (ADC) as well as mixed-signal sensors such as time-domain configurable analog modules and time-based resistive sensor interfaces [7, 8].

To address such a growing demand and deliver the missing design automation element, this paper lays out a systematic design automation flow for TC. More specifically, a digital compatible synthesis and backend flow is developed with novel variation aware RTL mapping and ACG-based placement algorithm to enable the automation of TC design. The proposed scheme is compared with existing analog placement and commercial EDA tool showing significant improvement in the matching performance. A test chip is used to show the satisfactions of design specification, e.g. mismatch, using the proposed digital compatible design flow.

2 TIME-DOMAIN COMPUTING (TC)

2.1 Time-Domain Computing Overview

TC encodes information into the time or delay of digital circuits and perform computation in time domain [3-6]. The system normally consists of time encoders for digital to time conversion, time-domain logic modules and optional time decoders for time to digital conversion as shown in Fig. 1.

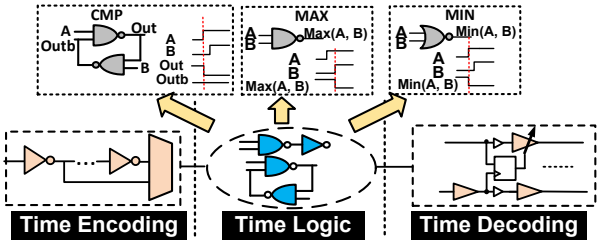


Figure 1: Overview of time-domain computing.

To facilitate the time-domain logic operation, a set of "standard cell" like modules are built for operation in time domain. One of the key advantages for TC is that all the building blocks are digital modules making the whole design digital friendly. Examples of TC circuits are also shown in Fig. 1.

2.2 Challenges of Time-domain Computing

As TC relies on the precise timing control for information processing, variation and mismatch of signal timing could lead to computation errors. As the least-significant-bit (LSB) resolution is pre-defined, e.g. 25*ps* used in this work, a variation of timing beyond this value will lead to single-bit error. Specially, local

DAC '19, June 2–6, 2019, Las Vegas, NV, USA

ACM ISBN 978-1-4503-6725-7/19/06…$15.00
https://doi.org/10.1145/3316781.3317800

variation or mismatch creates the largest threat to the operation similar to analog computing. Comparing to digital design, a much more stringent backend layout is needed in consideration of matching, variation, cross-talk and signal slew rate. In addition, as TC usually performs more complex algorithms [3, 4], the signal paths and matching components in TC are much more complicated than a typical analog design leading to more challenges in the front-end or back-end design for TC.

2.3 Proposed Digital Compatible Design Methodology

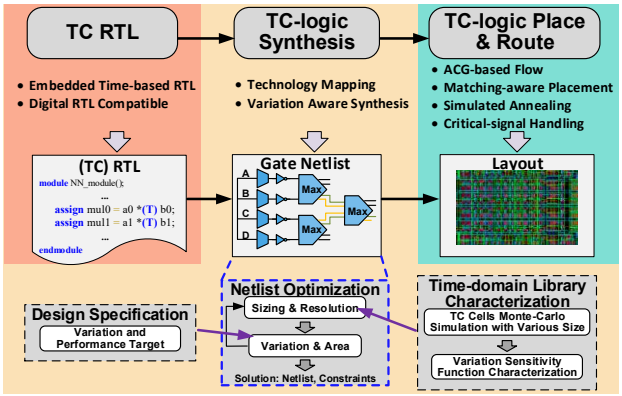


Figure 2: Flowchart of proposed TC automation flow.

Fig. 2 shows the overview of the proposed digital compatible design automation flow. Particularly, a specially developed time-domain RTL code is attached to conventional Verilog language to denote the special design of the time-domain logic operation. Based on the hybrid RTL codes, the synthesis tool provides logic synthesis and technology mapping to create a gate-level netlist using both standard cells and digital friendly time-domain modules. Variation awareness is implemented into the synthesis process. At the back-end, an ACG-based placement technique is developed to handle the stringent signal matching requirement of TC design.

3 SYTHESIS OF TIME-DOMAIN LOGIC

To create a digital-compatible design flow for TC design, synthesis has to be performed to create gate-level netlist similar to the conventional digital design. The proposed technique is realized by embedding a special plug-in script into existing RTL/synthesis flow. It handles not only the generation of time-domain cells but also special requirements in TC, such as variation.

3.1 Overview of Proposed TC Synthesis Technique

3.1.1 Special Synthesis Requirement in TC Design

Since the data is carried by the time or delay of the circuit cells, variation has large impact on the accuracy of design. Thus, minimizing the variation of the data path is quite critical for TC. The special design considerations of TC design are listed as:

1) Determining the size of the modules in TC circuit is a trade-off between area/energy consumption and variation/error-rate of the whole design. Increasing the size of a module can decrease the variation but increase the area and energy of the module.

2) The single-bit delay that represents "1" in digital-domain must be carefully chosen. Shortening the single-bit delay, can boost the performance but increase the error rate of the final result.

3.1.2 Proposed Synthesis Flow

The bottom of Fig. 2 shows the flow of the proposed synthesis technique: (1) the RTL with customized syntax for time-domain logics is utilized to perform a special TC logic synthesis process. As a result, both conventional digital and time-domain logics are synthesized into an initial gate-level netlist. The size of each cell is set to the smallest size at this step. (2) The initial netlist is then sent to a netlist optimizer to exercise the sizing options of each module to meet the variation budget while minimizing area consumption.

3.2 Implementation of TC Synthesis

3.2.1 Special Logic Mapping in TC Design

The proposed logic synthesis script can recognize special syntax used for the TC RTL. In the TC RTL, a special syntax is developed to denote the TC operation, e.g. add and multiplication. The special keyword "(T)" after the operation symbol "+" or "×" is used to denote the TC operation as shown in Table 1. The synthesis script works as a plug-in script on top of conventional synthesis tool. Special mapping functions are called for generating time-domain circuits similar to the conventional technology mapping. For instance, the "?" operation symbol in time-domain RTL, is mapped into a time-domain comparator which was shown in Fig. 1.

3.2.2 Variation Sensitivity Function

The variation sensitivity function is introduced for netlist optimization. We define the 3-sigma variation of TC modules, which is a function of the size s as $\sigma(s)$. Apparently, the $\sigma(s)$ decreases as s increases. The area of TC modules is a function of the size s as $A(s)$. The variation sensitivity function is shown as:

$$F_{sen}(s) = \gamma \frac{d\sigma(s)}{dA(s)} \quad (1)$$

where $\frac{d\sigma(s)}{dA(s)}$ term represents the variation sensitivity comes from the module, and γ term represents the significance of the module, e.g. module in a convergent path. As most TC cells are standard-cell like, we follow the standard cell sizing convention, e.g. 1X, 2X, etc.

3.2.3 Netlist Optimization

Assume that we have totally n modules, X_1, X_2, … X_n, the size of each module is $s_1, s_2, \ldots s_n$. Besides, there are p critical paths need to be considered in the placement. The optimization problem of netlist is then formed in (2) and (3):

$$\text{Minimize} \ \sum_{i=1}^{n} A(s_i) \quad (2)$$

$$\forall\, paths \in P, s.t. \sqrt{\sum_{i=1}^{n} {\sigma_p}^2(s_i)} \ \leq \ \sigma_T \quad (3)$$

where $\sigma_p(s_i)$ is the variation comes from X_i, and $A(s_i)$ is the area of X_i. The pseudo code of the optimization is shown as follows.

Algorithm 1 Netlist Optimization Algorithm

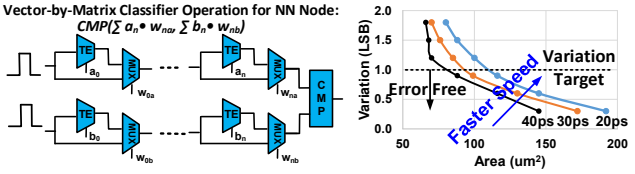
Input: Initial netlist of module X_1, X_2, …X_n, with minimum sizing $s_1, s_2, \ldots s_n$.
Output: Netlist which satisfies variation budget with minimum area

```
for all critical paths p in the netlist do
    while sqrt(sum_{i=1}^{n} σ_i^2(s_i)) > σ_T do
        for i = 1 to n do
            find the module j = i, with maximum F_sen(s_j)
        end
        Increase the size of module j by 1X, update s_j
    end
end
Return the netlist with current sizing
```

Given the initial netlist generated by TC logic mapping from TC RTL with minimum sizing, we first check if the variation of all critical paths meets the budget σ_T. If yes, the optimization is completed. Otherwise, the following step is performed in which we traverse the netlist to find out the most effective module in the critical path, i.e. highest variation sensitivity. The size s of this module is then increased by 1X. We keep repeating the previous steps until the variation targets of all critical paths are met.

3.2.4 Design Example on Time-domain Neural Network Module

An example of a simple TC neural network building block of vector by matrix classifier [5] is shown in Fig. 3 (a). The circuit contains 2 MAC and 1 CMP in time domain with RTL given in Table 1. The synthesised TC netlist with proper sizing after optimization is shown in Table 2. Fig. 3 (b) shows the design trade-off between the area and error rate (variation) by a given LSB resolution. Under different resolution and error tolerance, the optimal area of such a TC NN module is shown in Fig. 3 (b). With the same resolution, the area drops with the variation increases. For example, the areas are $195um^2$ and $110um^2$ with variation of 0.3 LSB and 1 LSB respectively when resolution is set to $20ps$.



(a) (b)

Figure 3: (a) Schematic NN module, (b) design trade-off between error tolerance and area with given single bit delay.

Table 1: Example RTL implementation of TC-neural node.

```
1   module NN_module (a0, a1, a2, a3, b0, b1, b2, b3, out);
2     input [1:0] a0, a1, a2, a3;
…                    …
6     assign mul0 = a0 *(T) b0;
…                    …
11    assign mac1 = mul2 +(T) mul3;
12    assign out = (mac0 >= mac1) ?(T) 0 : 1;
13  endmodule
```

Table 2: Example netlist of TC-neural node from synthesis.

```
1   module NN_module (a0, a1, a2, a3, b0, b1, b2, b3, in, out);
2     input [1:0] a0, a1, a2, a3;
…                    …
6     TC_TE_X3 I0 (.IN(in), .DIN(a0), .OUT(te0));
…                    …
13    TC_MUX_X4 I7 (.A(mul2), .B(te3), .S(b3), .OUT(mac1));
14    TC_CMP_X2 I8 (.a(mac0), .b(mac1), .out(out));
15  endmodule
```

4 PROPOSED MIXED-SIGNAL PLACEMENT

Due to the lack of prior techniques on automatic placement for TC circuits [3-6], in this section, we propose a practical and efficient placement technique for TC circuit utilizing adjacent constraint graph (ACG) based optimization engine to deal with the stringent matching requirements. It is worth to mention that although automatic placement has been proposed previously for analog/mixed-signal design [9, 10], TC poses special challenges, i.e. massive-stage-symmetry (MASS), as referred in this paper, and hence requires special techniques not available from the prior work.

The special matching requirement of MASS for time domain circuits are highlighted as follows:

1) Module symmetry and stage symmetry constraint: modules within certain groups must be placed symmetrically with respect to a horizontal or a vertical axis to maintain the matching of critical TC signal. Moreover, modules on symmetry paths need to be place symmetrically in each stage.

2) Clustering constraint: certain TC modules must be placed near to each other in order to isolate the critical TC modules from other digital modules.

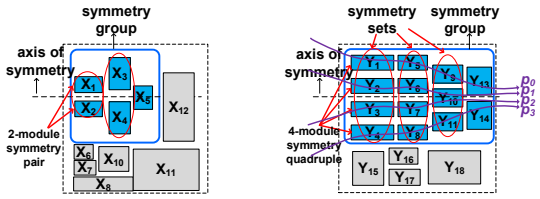
3) Shortest critical signal path constraint: the wire length of critical paths must be minimized in order to relieve the variation impact of TC circuit and improve slew rate of the signals.

Similar constraints are observed in the existing analog placement/floorplan design, but TC design has more challenges due to its larger numbers of components as described in the follows.

4.1 Preliminaries

4.1.1 Comparison with Previous Analog Placement Work

Topological representations are widely used in solving analog placement problems, in which, the relative positions between the modules are encoded. Typical topological representations are slicing tree [11], sequence-pairs (SP) [12], O-tree [13], B*-trees [14], and TCG-S [15]. Most of these works have been applied to handle the symmetry constraint and other constraints like the centroid constraint. However, these representations are not suitable for solving the MASS placement problem of TC design as explained as follows.



(a) (b)

Figure 4: Symmetry group in (a) conventional analog design, (b) time-domain computing design.

1) A complete representation is preferred in order to efficiently handle the special constraints like symmetry and critical path constraints. For example, tree-based representation doesn't provide complete topological information, which makes it harder to check the relations, e.g. horizontal relation, between modules.

2) When dealing with symmetry constraint, we form a symmetry group with multiple symmetry pairs. However, in most of analog placement problem, each symmetry pair in the symmetry group only contains few modules as shown in Fig. 4 (a). On the other hand, in the TC design, large numbers of modules, defined by the algorithm, e.g. LDPC [3], need to be allocated symmetrically through hierarchies as shown in Fig. 4 (b).

3) For TC design, we not only need to place the modules symmetrically within a set, but also need to guarantee the matching across different hierarchy on the long signal paths. As shown in Fig.4 (b), the modules on path p_0 must be symmetric with the modules on paths $p_1 - p_3$ leading to stringent multi-path matching problems for sequence of modules. This not only requires a massive symmetry placement within a symmetry group but also requires carefully match at each stage. Thus, the MASS becomes a special challenge in the TC placement.

Adjacent Constraint Graph (ACG) [16] representation is chosen in this work due to the following advantages: compared with existing placement techniques, ACG has the advantage of efficiency and succinctness when dealing with the symmetry and other constraints. Without the redundant edges, the number of edges in ACG is O(nlog(n)), much smaller than the $O(n^2)$ number of edges in TCG-S or SP. ACG is also more flexible than other representations in performing packing.

4.1.2 Problem Formulation

Assume we are given a set of n modules with areas A_i where $i = 1 \ldots n$, together with a set of j nets $N_1, N_2 \ldots N_j$. Our objective is to obtain a placement F of the circuit satisfying all the placement constraints mentioned previously while minimizing a cost function:

$$C(F) = A(F) + \alpha \times W(F) + \beta \times W\_penalty(F) \quad (4)$$

where A(F) is the total area of F, W(F) is the total wire length of F, W_penalty(F) is the total wire length of wires between the modules which violated the constraint after the packing stage. α and β are empirical coefficients used for regulating the weights of wire length and wiring violation.

4.2 Adjacent Constraint Graph (ACG) Representation

The basic idea of the ACG representation, briefly described below, is to encode any rectangle packing as an ordered module sequences with edges which indicates the spatial relations [16].

As an illustration, for a floorplan given in Fig. 5 (a), its constraint graph in both horizontal and vertical directions are shown in Fig. 5 (b). As the essential idea of constraint graph is used for avoiding module overlap, any two modules must have at least one relation ("left" or "below to"). Thus, over-specification has no benefit in terms of representation. Since those redundant edges are unnecessary for placement, we can remove those edges and the result is an ACG representation (Fig. 5 (c)). The corresponding ACG data structure is shown in Fig. 5 (d). The vertices will be doubly linked in a linear order. Edges are all directed from left to right. The edges above the vertex line represent horizontal (H) relations and those below represent vertical (V) relations.

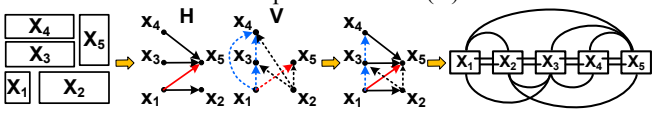


Fig. 5. (a) A floorplan, (b) constraint graphs in horizontal (solid edges) and vertical (dotted edges) directions, (c) ACG Graph, (d) ACG data structure.

4.3 Proposed TC Placement Approach

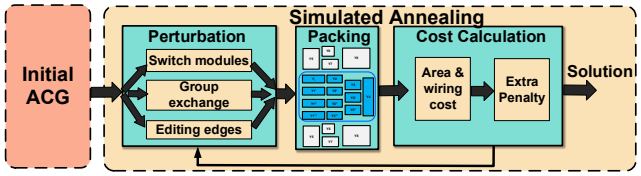


Figure 6: Flowchart of the proposed placement.

Simulated annealing is employed as the basic searching engine in our approach with ACG as the representation. Our proposed placement algorithm works as follows. It first generates an initial ACG representation following the default cells order, which also satisfies all the constraints proposed by the designer. After the initial solution is generated, the simulated annealing process is applied. In each iteration the following steps are performed: (1) three categories of perturbations/moves are introduced. All these perturbations are complete in terms of the searching space; (2) After the perturbation, a new ACG is generated and the corresponding packing is produced based on the longest path algorithm; (3) Area and interconnect cost with extra penalties are computed based on the new packing. (4) Check whether the annealing process should continue based on the current temperature and cost. The flowchart is shown in Fig. 6.

4.4 Handling of Placement Constraints in TC

4.4.1 Handling of Symmetry Constraint

In TC circuit, symmetry constraint (marked in blue in Fig. 7 (a)) can be handled as follows (we assume the symmetric modules are symmetric with respect to a horizontal axis):

1) If modules Y_1, Y_2, Y_3, and Y_4 are required to be symmetric, all of them must be in vertical relations. In the other word, every two of them must be connected by horizontal edges in the ACG.

2) The x coordinates of modules Y_1, Y_2, Y_3, and Y_4 must be same which can be regulated during the packing stage.

3) The distances between adjacent modules must be same.

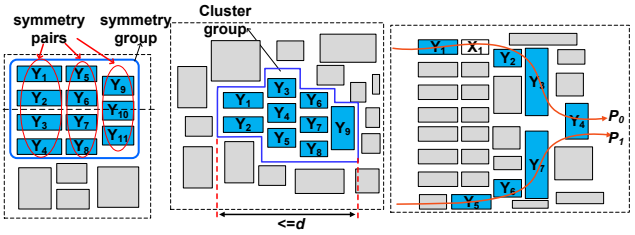


Figure 7: Example of (a) symmetric constraint, (b) clustering constraint, (c) critical signal path constraint.

4.4.2 Handling of Clustering Constraint

Clustering constraint can be handled by forcing the modules in the same clustering group to abut each other in ACG representation. Besides, we introduce the penalty term in the cost function to force the placement to obey the constraint. An example of clustering constraint among modules Y_1-Y_9 is shown in Fig. 7 (b).

4.4.3 Handling of Critical-Signal Path Constraint

To handle this constraint, the total wire lengths of these paths need be as short as possible (P_1 and P_0 in Fig. 7 (c)). The constraint can be handled by (1) guaranteeing horizontal relations for the modules in same critical path in ACG, e.g. Y_1, Y_2, Y_3 and Y_4; (2) increasing the weight of nets which are on the critical paths when calculating the cost of total wire length. As a result, the placement engine tends to move the modules which are not on critical signal path, e.g. X_1, away from the critical path P_0.

4.5 Set of Perturbations/Moves

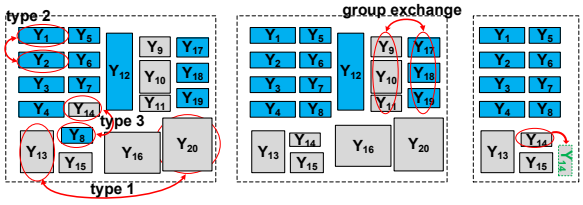


Figure 8: Example of moves in (symmetry group are marked in blue): (a) category 1, (b) category 2, (c) category 3.

We employ the following set of moves to perturb a current candidate ACG. The moves/perturbations can be divided into three categories: (a) exchange of two random modules, (b) group exchange of the symmetric sets, and (c) editing edges in the current ACG representation. The details of moves are given as follows:

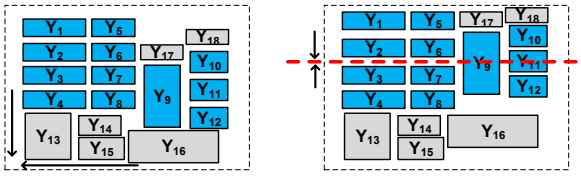
1) In the first category (Fig. 8 (a)), there are three different types of exchanges: (1) Exchange two random modules which are not in any of the symmetry groups. (2) Exchange two random modules within a symmetric set. (3) Exchange one module which is inside of one symmetry group and another module which is outside of that symmetry group. This movement cannot be guaranteed to not violate the symmetry constraint. Thus, a special checker is implemented to check the feasibility of the new generated ACG. If such a move violates the constraints, penalty will be added to the cost function shown in eq. (4).

2) Fig.8 (b) shows one example of second category. This group exchange also needs special checker to check the feasibility of the new ACG after such a move. It provides the chance of moving away the modules which are located inside of a symmetry group.

3) The third category involves the modification of ACG edges including (1) changing current edge type from horizontal to vertical or vice versa; (2) Adding or removing the existing current edges while following the ACG requirement. We only allow modifying the edges of the modules which are outside of symmetry group. In this way, all the constraint within the symmetry group cannot be violated. An example of modify the edge between Y_{14} and Y_{15} from vertical to horizontal is shown in Fig. 8 (c).

4.6 Packing and Routing

A new packing algorithm is derived from conventional packing scheme based on the longest path algorithm. Different from previous work, the proposed packing algorithm allows us to pack the selected modules in respect to the symmetry axis instead of only to the lower bottom corner of plane [12, 14]. The packing example of conventional and our proposed ways are shown in Fig. 9 with symmetric modules marked in blue.



(a) (b)

Figure 9: Example of packing (a) to lower-bottom corner, and (b) respect to the symmetry axis.

We utilize the Innovus tool to handle the routing job. Since the TC cells follow the digital cell's implementation and are well organized after the proposed placement, e.g. the cells on the same critical path are placed abut to each other, the Innovus tool can handle the routing job appropriately. However, we expect more sophisticated routing methods to be developed for larger TC design as a future work.

5 EXPERIMENTAL RESULTS

5.1 Time-domain WTA Operation Implementation

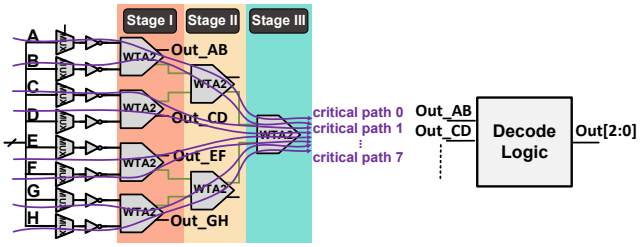
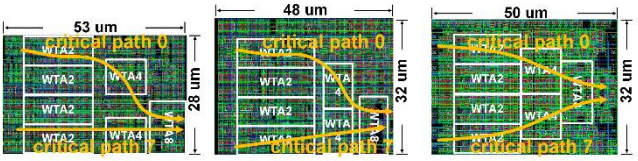


Figure 10: Topology and implementation of WTA in TC.

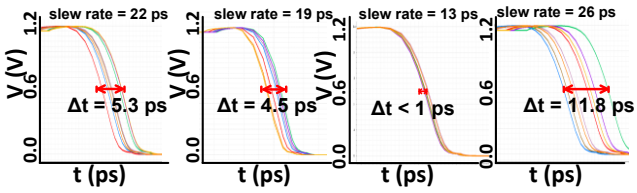
We compare our proposed ACG-based placement flow to other existing work [12,14] on a winner-take-all (WTA) circuit, which is a commonly used digital module in machine learning based classifiers. Fig.10 shows the design of the 8-input 6-bit WTA. The algorithm of WTA is based on binary comparison tree. The critical signals are propagated through 3 stages and the matching of 8 critical paths is the key concern of the design. The total number of critical digital modules for matching are 84 which is much larger than a typical matching problem observed in an analog design.

We experiment the placement of WTA by different approaches: (a) use B* tree based placement method from [14], (b) use sequence pair (SP) based placement method from [12], (c) use the proposed placement method. The layout results of approaches (a), (b) and (c) are shown in the Fig. 11. All the methods maintain a good symmetry property in the 1st stage (WTA2). However, both B* tree based and SP based placement methods have troubles in placing the modules properly in the stages 2 and 3 as (1) the modules in 2nd and 3rd stages are not placed in the central region with respect to the 1st stage leading to large signal routing mismatch between critical signals; (2) The critical TC modules are not separated with other non-critical modules causing the slew rate degradation of the critical signals. These failures are mainly due to the following reasons: (1) both previous placement approaches pack the modules from lower bottom corner leading to difficulty in placing the selected modules in respect to the symmetry axis; (2) Both previous placement methods are short of the ability to deal with the clustering and critical-path constraints. As a result, they failed to place the critical time-domain modules to be close to each other avoiding non-critical modules to block the critical paths. On the other hand, due to the efficiency and succinctness of ACG-based representation, it's much easier to handle the cluster and critical path constraints. As a result, the above issues can be properly resolved by the proposed ACG-based placement with good matching through stages of critical paths (Fig.11 (c)).



(a) (b) (c)

Figure 11: Layout of placement methods: (a) B* tree based [14], (b) sequence pair based [12], (c) proposed design in this work.



(a) (b) (c) (d)

Figure 12: Simulation result of mismatch for (a) B* tree based placement [14], (b) sequence pair based placement [12], (c) our proposed technique, (d) conventional digital design.

After the layout is generated from Innovus, we import the layout back into Cadence Virtuoso to perform spice simulation with parasitic extraction. The simulation result of matching for the 8 critical paths is shown in Fig 12 in comparison among B* tree method, SP method, proposed method and conventional digital design using EDA tools. As we can see, the mismatch from using B* tree based and SP based placement method are better than that from the conventional digital flow. However, the mismatch from these two methods are still significantly larger than our proposed ACG-based placement method whose mismatch is less than 1*ps*. Thus, the proposed placement methodology provides both the efficiency and accuracy in dealing with TC design. Table 3 summarizes the performance of different methods. The algorithms are implemented in C++ and run on a Windows machine with 2.6GHz i7 Quad-core and 8GB RAM. Note that ACG-based

placement method also achieves the lowest runtime mainly due to the efficient and succinct representation when deal with complex matching constraints. For example, the number of edges in ACG is O(nlog(n)), while it's $O(n^2)$ in SP. Even though the edge number is only O(n) in B* tree, it lacks a complete topology information used for dealing with TC constraints which makes the number of searching iteration larger.

Table 3: Performance Comparison for Placement Methods.

Methods	B* tree [14]	SP [12]	This work
Mismatch (ps)	5.3	4.5	1
Slew rate (ps)	22	19	13
Run time (s)	23	85	18
Area (um^2)	1484	1536	1600

5.2 Time-domain Image Processing Implementation

For demonstration, we adopt a basic facial recognition algorithm into a hybrid ASIC design with time-domain accelerators. The operations of the image recognition algorithm involve three steps: (1) feature extraction which performs median filtering and detects edges in four directions. (2) Vector formation; (3) Classification where the generated feature vector is classified by a winner-take-all (WTA) classifier. In our design, the median filter for feature extraction and WTA for final classification were designed in time-domain to remove the bottlenecks of the algorithm [17]. In particular, the proposed synthesis and placement techniques were applied on the WTA design leading to the layout for the fabricated chips.

5.3 Measurement Results

The 55nm test chip was fabricated and measured across 10 chips. No error was observed at internal time-domain results or final classification at the design target speed of 1.33GHz.

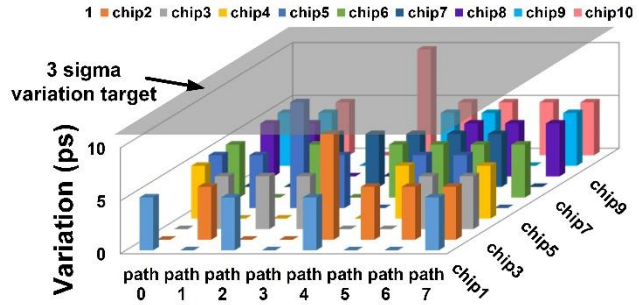
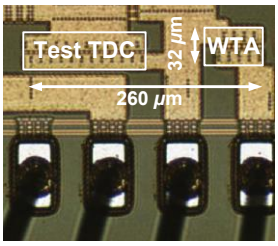


Figure 13: Mismatch measurement results; y axis denotes the absolute variation from the nominal delay.

Fig. 13 shows the measured on-chip mismatch of 8 critical paths from 10 chips in WTA circuits. The mismatches were measured by using an on-chip time-digital-converter (TDC) with 5ps resolution. As shown, the measured mismatch is within 0.5 LSB, which verifies the feasibility of handling variation (synthesis) and layout mismatch (placement) of the proposed methodology. No systematic mismatch was observable from the measurement proving the good matching performance of the placement algorithm. The mismatch was dominated by the random process variation which has been properly budgeted (within half of LSB, i.e. 12ps as 3-sigma variation target) from the proposed synthesis flow. The die micrograph and the specification of WTA is shown in Fig. 14. The design is compared with conventional ASIC with standard synthesis and place and route implementation. A 42% area saving, a 1.7X speedup and a 23% power saving, is observed in the time-domain WTA accelerator compared to ASIC implementation. The overall image recognition processor operates at 1.33GHz with a state-of-art throughput of 72 frames per second.



Technology	55 nm	
Frequency (GHz)	1.33	
Total Chip Area (mm^2)	0.64	
	ASIC	TC
WTA Area (μm^2)	2800	1600
WTA Power (mW)	3.1	2.4
WTA Frequency (GHz)	1.2	2

Figure 14: Die photo and specifications of the WTA design.

6 CONCLUSION

This paper proposed a comprehensive digital compatible design flow including front-end synthesis and backend placement for TC. In the synthesis stage, our proposed technique can handle the variation requirement while minimizing the estimated area of the circuit. In the backend stage, an ACG-based placement algorithm is developed to handle the complex placement constraints for TC design. The comparison with prior analog placement schemes shows much improved matching performance from the proposed method. The proposed synthesis and placement flow is demonstrated by a 55nm test chip showing on-target mismatch results and significant performance enhancement from TC compared with digital implementation.

REFERENCE

[1] Yong Shim, et al, "Low-Power Approximate Convolution Computing Unit with Domain-Wall Motion Based "Spin-Memristor" for Image Processing Applications", *IEEE/ACM DAC*, 2016

[2] F. N. Buhler, et al, "A 3.43TOPS/W 48.9pJ/Pixel 50.1nJ/Classification 512 Analog Neuron Sparse Coding Neural Network with On-Chip Learning and Classification in 40nm CMOS", *VLSI Symposium*, 2017.

[3] Daisuke Miyashita, et al, " An LDPC Decoder With Time-Domain Analog and Digital Mixed-Signal Processing", *IEEE JSSC*, 2014.

[4] Anvesha Amravati, et al,"A 55nm Time-Domain Mixed-Signal Neuromorphic Accelerator with Stochastic Synapses and Embedded Reinforcement Learning for Autonomous Micro-Robots", *ISSCC*, 2018.

[5] Daisuke Miyashita, et al, "Time-Domain Neural Network: A 48.5 TSOp/s/W Neuromorphic Chip Optimized for Deep Learning and CMOS Technology", *IEEE ASSCC*, 2016.

[6] M. Liu, et al, " A Scalable Time-based Integrate-and-Fire Neuromorphic Core with Brain-Inspired Leak and Local Lateral Inhibition Capabilities", *IEEE CICC*, 2017.

[7] Yunju Choi, Yoontaek ,Seung-Heon Baek, Sung-Joon Lee, Jaeha Kim, "A Field-Programmable Mixed-signal IC with Time-domain Configurable Analog Blocks", *IEEE Symposium on VLSI Circuits*, 2016.

[8] Jorge Marin, E. Sacco, J. Vergauwen,Georges Gielen, "A Single-Temperature-Calibration 0.18-µm CMOS Time-Based Resistive Sensor Interface with Low Drift over a− 40° C to 175° C Temperature Range", *IEEE ESSCIRC*, 2018.

[9] Lin, P.H., et al, "Analog Placement Based on Hierarchical Module Clustering", *IEEE/ACM DAC,* 2008.

[10] Biying Xu, et al, "A scaling compatible, synthesis friendly VCO-based delta-sigma ADC design and synthesis methodology", *IEEE/ACM DAC*, 2017.

[11] Chang-Tzu Lin, et al, "An Efficient Genetic Algorithm for Slicing Floorplan Area Optimization", *IEEE ISCAS*, 2002.

[12] Qiang Ma, et al, "Simultaneous Handling of Symmetry, Common Centroid, and General Placement Constraints", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2011.

[13] P.-N. Guo, et al, "An O-Tree Representation of Non-Slicing Floorplan and Its Applications*," IEEE/ACM DAC*, 1999.

[14] Pang-Yen Chou, et al, "Heterogeneous B*-trees for Analog Placement with Symmetry and Regularity Considerations," *IEEE ICCAD*, 2011.

[15] J.-M. Lin and Y.-W. Chang, "TCG-S: Orthogonal Coupling of P*- Admissible Representations For General Floorplans," *IEEE Trans. CAD*, 2004.

[16] Hai Zhou and Jia Wang, "ACG–Adjacent Constraint Graph for General Floorplans", *IEEE ICCD*, 2004.

[17] Z. Chen, et al, "A Time-Domain Computing Accelerated Image Recognition Processor With Efficient Time Encoding and Non-Linear Logic Operation," *IEEE JSSC*, Nov. 2019.