CINEFORUM
ISSN: 0009-7039
Vol. 66. No. S6, 2026 (Special Issue 6)

TriNet-Based Hardware-Aware Deep Learning Framework for Automated Analog Circuit Parameter Prediction

Bommakanti Meghana[1*], Dr.I.V. Prakash[2*], Dr. M.V. Raghavendra[3*], D.Satheesh[4*]

[1]Student of M.Tech in VLSI System Design, Department of Electronics and Communication Engineering, Sree Dattha Institute of Engineering and Science, Hyderabad, Telangana, India.
[2]Assistant Professor, Department of Electronics and Communication Engineering, Sree Dattha Institute of Engineering and Science, Hyderabad, Telangana, India.
[3]Associate Professor, Department of Electronics and Communication Engineering, Sree Dattha Group of Institutions, Hyderabad, Telangana, India.
[4]Assistant Professor, Department of Electronics and Communication Engineering, Sree Dattha Institute of Engineering and Science, Hyderabad, Telangana, India.
Corresponding Author
Email: meghanaiyengar31@gmail.com

Abstract

Analog integrated-circuit design is less efficient than virtual design due to the high non-linearity and coupling between performance specifications and device measurements and passive-component values. A hardware-aware TriNet framework is shown in this paper that predicts the parameters of analog circuits directly using design specification. Prediction is done by first normalizing and analyzing the input vector, which includes specifications: gain, bandwidth, phase margin, slew rate, and offset, with a feasibility classifier. Three sequential learners are used to process their valid input: a basic learner refines coarse mapping, an intermediate refiner is used to refine further and an advanced learner can be used to refine even higher order nonlinear correction. Their products are stored and converted back to physical quantities using postprocessing. It is formulated as an architecture and is implemented as fixed-point Verilog and comprises deterministic control, blocking of invalid specifications, and scalable fully connected inference modules. The suggested strategy should decrease the design turnaround time and simulation reliance and maintain physical feasibility. The authors intentionally insert experimental results with data of their synthesis, simulations, forecasts.
Keywords—Analog design automation, circuit parameter prediction, Deep learning, TriNet, residual learning, Verilog, fixed-point inference, feasibility classification.

1. INTRODUCTION

Even at the analog and mixed-signal level, variation in a transistor dimension, bias current or resistance or capacitance can affect gain, bandwidth, noise, stability, power, and output swing in multiple ways, and therefore is hard to automate. Traditional design thus follows the process of successive sizing, simulation, interpretation and correction. This is precise but time consuming and depends on the designer, and requires a high level of computation, as it may require that many process, voltage and temperature configurations are checked [1] -[5].

Analog design by machine-learning can substitute some of this search by learning an inverse mapping between desired performance and circuit parameters. After being trained on simulated data of representative runs, a given predictor can provide a high-quality starting point, or even directly determine device sizes. However, a single network can fail to model the full inverse relationship especially when the consequent region is broken or multiple specifications are opposed to each other. Another thing is that a predictor can give physically meaningless values when a requested specification vector is in the training and realizable design space.

In this work, a structured TriNet predictor is developed where three learners gradually create a coarse, residual and complex nonlinear model of behavior. The model is made predictor-suitable by a feasibility classifier, and by fixed-point preprocessing and postprocessing to be a synthesizable Verilog inference engine. This contribution is not a neural model, but a full-fledged hardware-conscious architecture that bridges user specifications and deployable circuit parameters to explicit valid control.[6],[7].

Its key contributions include: (1) an architecture of phase-wise progressive learning of specification-to-parameter regression; (2) an explicit feasibility gate that blocks invalid specifications; (3) fixed-point formulations of normalization, dense-layer evaluation, accumulation of output, and inverse scaling; and (4) a Verilog-oriented modular inference flow that can be linked to analog design automation environments.

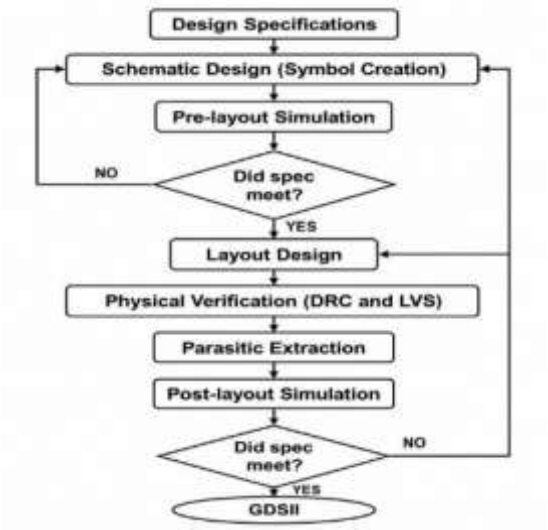

Fig.1. Conventional analog IC design flow involving repeated pre-layout and post-layout verification.

2. RELATED WORK

Historical approaches to analog synthesis were based on analysis equations and user-specified constraints to start with device sizes. Geometric programming enhanced the solution of certain convex transistor-sizing issues, and realistic analog behaviour comprises nonlinearities, parasitics, discrete decisions, as well as trade-offs that are not necessarily convex, which can be difficult to represent in simplified models [4], [5]. The simulation methods enhanced fidelity by considering SPICE as the assessment engine, but by repeatedly calling the simulator, consumed significant time by the simulator.

Stochastic optimization algorithms, such as genetic algorithms, particle swarm optimization or differential evolution are able to search spaces that are non-convex and do not need derivatives [7]-[10]. Cost of evaluation is their main weakness: each candidate is typically in need of a circuit simulation. Moreover, a new optimization run is very often launched with each new specification set, therefore, the computation is not efficiently reused.

Supervised regression can be reused by exploiting previously simulated designs. Performance estimation and design-space modeling have used linear models, support vector machines, decision trees, random forests and multilayer perceptions. Deep neural networks are capable of making nonlinear mappings in high dimensions, but the level of accuracy is very dependent on the coverage of the data used in training. They are also able to extrapolate in an unpredictable way of what lies outside the feasible region.

The residual and ensemble learning enhance prediction by learning several models or by learning to perform the errors that are left by the previous stages. The proposed TriNet concept uses the same principle to analog parameter prediction: a shallow learner forms a shallow prediction, a second learner forms a residual structure, and a third, deeper learner forms a fine nonlinear correction. The proposed architecture, as opposed to an unconfined ensemble, has input feasibility classification, and a deterministic accumulation mechanism that can be implemented in hardware.

3. PROBLEM FORMULATION

Assume that the analog performance specifications of interest may be expressed as an input in Rm. Characteristics typical in a typical vector are DC gain, unity-gain bandwidth, phase margin, slew rate, input offset and power, common-mode rejection ratio, power-supply rejection ratio and output swing. The designed n-vector y R n can include transistor widths and lengths, bias currents, capacitance values of compensation, and load capacitance and resistance.

$$\mathbf{x} = [x1 \quad x2 \quad \cdots \quad xm]^T, \mathbf{y} = [y1 \quad y2 \quad \cdots \quad yn]^T(1)$$

The inverse-design problem Inverse-design the task is to learn the target function F which approximates y given x (with feasibility constraints). Because not all specifications vectors are practical with the topology and technology of choice, a binary feasibility function C(x) is defined.

$$\hat{\mathbf{y}} = F(\mathbf{x}), \text{subject to } C(\mathbf{x}) = 1, C(\mathbf{x}) \in \{0,1\}(2)$$

Training reduces the gap between the predicted and reference parameters on a dataset D = {(x(k), y(k))). When the different output parameters take different scales, a normalized mean-squared goal is appropriate

$$L = \frac{1}{N} \sum_{k=1}^{N} \| D^{-1}(\hat{\mathbf{y}}^{(k)} - \mathbf{y}^{(k)}) \|^2$$

In which D long is a diagonal scale matrix. The non-negativity and device bounds or topology sensitive constraints can be enforced with extra penalties. In inference, the first step the classifier will take is to identify whether the request is within the valid design space. Illegal requests raise an error flag and turn off the output of parameters.

4. PROPOSED TRINET ARCHITECTURE

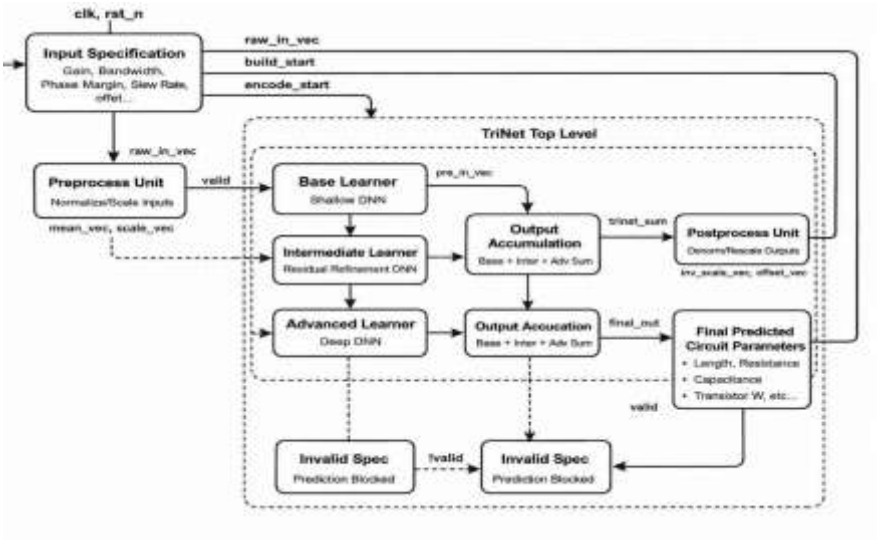

Fig.2.Proposed TriNet-based Verilog inference architecture for analog circuit parameter prediction.

4.1 Input Representation and Preprocessing

Because the specifications are expressed in different units and ranges, direct fixed-point inference can be numerically unstable. Each input is normalized using stored offset and scale constants. For feature i,

$$z_i = (x_i - \mu_i)s_i \quad (4)$$

where μ_i is the reference mean or offset and s_i is the reciprocal scale. For Q-format hardware with F fractional bits, the multiplication is implemented using an arithmetic right shift:

$$z_{i,\mathrm{fx}} = [(x_{i,\mathrm{fx}} - \mu_{i,\mathrm{fx}}) \times s_{i,\mathrm{fx}}] \ggg F \quad (5)$$

The preprocessing unit stores constants in registers or ROM and produces a normalized vector together with a valid strobe. Saturation logic is recommended to prevent overflow when input values exceed the trained range.

4.2 Feasibility Classifier

The feasibility classifier rejects specification combinations that are outside the realizable region. It may be implemented as a compact decision tree, a shallow neural classifier, or a set of learned threshold rules. In the simplest bounded form,

$$C(\mathbf{x}) = \bigwedge_{i=1}^{m} [l_i \le x_i \le u_i] \quad (6)$$

where l_i and u_i denote feature limits. A learned classifier is preferable when feasibility depends on coupled conditions rather than independent bounds. The classifier output controls the enable signal of all three prediction learners. This avoids meaningless estimates and reduces unnecessary switching activity.

4.3 Base Learner

The base learner is a shallow fully connected network that models the dominant inverse relationship. For layer ℓ,

$$h_b^{(l)} = \phi_b^{(l)}(W_b^{(l)} h_b^{(l-1)} + b_b^{(l)}), \; h_b^{(0)} = z_b \quad (7)$$

where φ(·) is typically ReLU. Its output is a coarse estimate y_β. A shallow structure is intentionally used to learn smooth, high-energy trends without forcing one network to represent every local interaction.

4.4 Intermediate Learner

The intermediate learner concentrates on mistakes that are not taken into account by the base learner. The residual r 1 y y − y 1 can be taken as the target in staged training. To model moderate nonlinearities, the learner can take into account additional hidden units and an ELU-like activation.

$$y_i = F_i(z; \theta_i) \approx y - y_b \quad (8)$$

Hardware In hardware nonlinear activations can be computed using piecewise-linear approximations, small look-up tables, variants of ReLU. Increasing the selected hidden features with the original normalized input allows to retain low-level specification information and enhances the gradient flow in training.

4.5 Advanced Learner

The high-order residual that is remaining is estimated by the advanced learner. It is the farthest branch and may have numerous concatenation paths or left over links. It targets training in the following way:

$$y_a = F_a(z; \theta_a) \approx y - (y_b + y_i) \quad (9)$$

This decomposition splits the complexity of a model over three learners. It also allows the independent word-length and resource optimization: a base branch may be more generous in terms of precision of the more common terms, and the subsequent residual branches may be more sparse since their output will be smaller.

4.6 Output Accumulation and Postprocessing

Normalized TriNet prediction is calculated through the element-wise accumulation:

$$\tilde{y} = y_b + y_i + y_a \quad (10)$$

The language of the postprocessing unit reinstates the physical units through scale, target-specific constants of scale, and offset:

$$\hat{y}_j = \frac{\tilde{y}_j}{s_{y,j}} + \mu_{y,j} \quad (11)$$

Fixed-point multiplication is applied by the inverse target scale, rounding and saturation to hardware implementation. Issuing of the final vector only occurs when the classifier, learner pipelines, and the postprocessing stage is valid. Otherwise, a signal invalid-specification will be asserted and the output cleared or held as specified in the interface.

5 VERILOG-ORIENTED HARDWARE IMPLEMENTATION

5.1 Modular Organization

It is possible to subdivide the entire inference engine into input registers, a preprocessing unit, feasibility classifier, three learner modules, two accumulation stages, postprocessing logic and a controller. Every learner comprises of parameter memory, multiply-accumulate units, addition of uncertainty, activation, and pipeline registers. A typical dense-layer module can be reused with other dimensions and weight files and provides less verification effort.

Module	Function
input_spec_if	Captures specification vector and start handshake.
preprocess_unit	Applies mean subtraction, scaling, saturation, and range checks.
validity_classifier	Produces valid_spec/invalid_spec decision.
base_learner	Computes the coarse parameter estimate.
intermediate_learner	Computes first residual correction.
advanced_learner	Computes higher-order residual correction.
output_accumulator	Adds learner outputs with extended precision.
postprocess_unit	Restores engineering units and clips to legal ranges.
control_fsm	Sequences loading, inference, and output-valid signaling.

Table.1.Functional modules of the proposed inference architecture.

5.2

Fixed-Point Arithmetic

Fixed-point Implementation Minimizes the area and power of floating-point implementation. The choice of word length has to be made among the dynamic range measured during trained-model calibration. The products have dominant factors of about two times the operand width of the product themselves and guard bits that are another fan-in ratios in accumulated. A K-input dense neuron can be assessed as

$$a = b + \sum_{k=1}^{K} w_k \, z_k, h = \phi(a) \quad (12)$$

Truncation to maintain systematic bias should be rounded off. The reason is that saturation is desirable instead of wrap-around since otherwise the overflow would cause considerable nonphysical errors. Finite precision quantization-aware training can also mitigate the accuracy loss due to finite precision.

5.3 Control and Timing

IDLE, LOAD, Preprocess, CLASSIFY, BASE, Intermediate, ADVANCED, Accumulat, Postprocess and Done are the states that can be used in a finite-state machine. Fully parallel design has least latency and maximum resource usage. A time-multiplexed multiply-accumulate array is a large area reduction, at the price of extra cycles. The architecture selected must be suitable to the necessary throughput of the target design assistant or EDA integration.

5.4 Training-to-Hardware Workflow

The workflow steps are advised to be: create a simulation dataset; partition it into training, validation and test partitions; normalize inputs and targets; train the classifier and three learners; quantize and test the model; analyse weights, biases and offsets and scales; load the constants into the Verilog memories; and compare the register transfer level results with the software golden model. The traceability of such conversion is necessary due to the fact that training accuracy is not necessarily sufficient to have correct neighboring fixed-point hardware inference.

6 EXPERIMENTAL METHODOLOGY

Here the part that demarcates the evaluation process but reserves to the authors all the numbers. The chosen analog topology and technology must be used to produce the data set by parameter sweeps that are controlled or space-filling sampling. All the designs of candidates are simulated to derive performance specifications. Samples that are not within operating-region, stability or design-rule constraints should be marked untrainable on the classifier.

6.1 Dataset and Splitting

Record the circuit topology, technology node, supply voltage, process corners, temperature range, the number of designs generated, the number of designs that are feasible, input specifications and projected parameters. Use mutually exclusive training, validation, and test sets. In the event of several process corners, stratified or corner split must be employed to prevent leakage.

6.2 Prediction Metrics

Mean absolute error, root-mean-square error, coefficient of determination, and average normalized error are some of the recommended measurements of regression. For output j,

$$\mathrm{ANE}_j = \frac{100}{N} \sum_{k=1}^{N} \frac{|\hat{y}_{k,j} - y_{k,j}|}{|y_{k,j}| + \varepsilon} \quad (13)$$

Some metrics that should be used to assess the feasibility classifier are accuracy, precision, recall, F1-score and a confusion matrix. False-accept rate is to be reported separately since even a design that is being propagated can be false due to an invalid specification that is declared to be valid.

6.3 Hardware Metrics

In the case of the Verilog implementation, report target FPGA or ASIC library, clock frequency,latency, throughput, lookup tables or gates, registers, DSP blocks, memory, dynamic power,static power and maximum operating frequency. Compare software floating-point inference, quantized inference, and RTL inference to check numerically the consistency.

6.4 Baseline Comparisons

TriNet predictor is to be compared to at least one single stage multilayer perceptron and (where possible) more conventional regression or optimization technique. The contribution of the feasibility classifier, the intermediate learner, the advanced learner and the output accumulation strategy should be quantified in an ablation study.

7 RESULTS AND DISCUSSION

The proposed TriNet diagram was written in Verilog, and tested with Vivado 2025.1. The provided deliverables are a behavioral simulation, a detailed RTL schematic, a synthesized-log, a synthesized-device view, and a power-estimation report. Such findings show that the representation can capture normalized analog-circuit

representations, process them by various stages of a learner, and produce ultimate prediction-control outputs in an expression of hardware. Simultaneously, the screenshots reveal that certain synthesis and power values are part of a testbench-based top module and thus cannot be considered the final resource and power values of the entire TriNet accelerator.

7.1 Behavioral Simulation Result

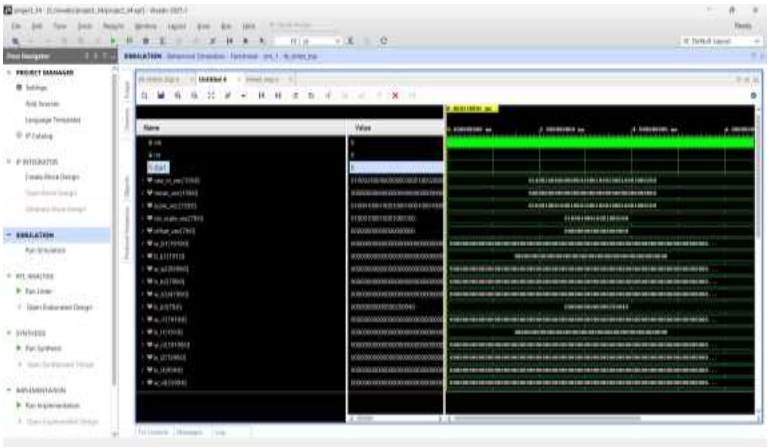

Fig.3.Behavioral simulation waveform of the TriNet prediction framework.

The behavioral simulation shows the clock, reset, start control, raw input vector, normalization parameters, inverse-scaling parameters and various weight and bias buses of the three-stage learning architecture. Each visible state is assigned a hexadecimal value and there are no mysterious (X) or high-impedance (Z) states in the interval shown. This means that the testbench manages to initialize the inputs and model parameters.

The source input feed is 160 bits and it is filled with a collection of the design variables in analog format expressed in fixed point. The average of the vectors is zero and the scale vector is not zero but has repeated factors. The inverse-scale and offset vectors are available as well to do post-processing. The base and intermediate and advanced learners weight and bias buses are set with deterministic values on the visible weight and bias buses, which attests that the hardware model has network parameters available to it during simulation.

At the chosen cursor position, the start signal is low, implying that the represented state is either before a new prediction request, or after the Pending transaction. Correct initialization and stable propagation of model data is however verified by the waveform; although prediction accuracy cannot be determined using this screen shot alone since last predicted parameter vector, valid pulse and error measure is not visible in the selected signal list.

7.2 Elaborated RTL Architecture

The detailed design has 196 cells and 535,725 nets. The huge net size corresponds to a hardware-optimized implementation of deep learning through open-ended feature, weight, bias, and intermediate activation vectors in features, weights, and post-processing, respectively. As represented in the schematic, the input section is dense with arithmetic and comparison stages, multiplexers, the registers and the last output-control logic.

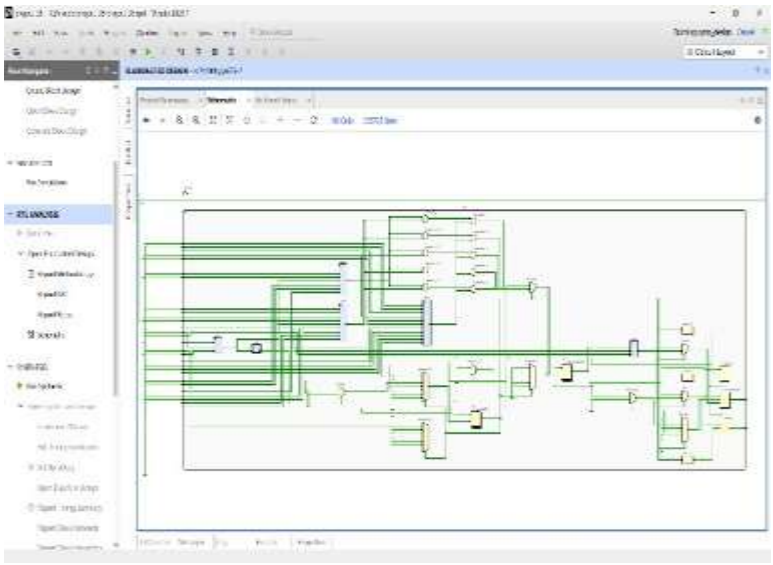

Fig.4.Elaborated RTL schematic of the TriNet hardware architecture

The architecture that is apparent in the schematic is of the anticipated TriNet flow. Raw specifications are accepted

and preprocessed and the initial estimate is created by the base learner, intermediate specifications optimize the result and the advanced specifications optimize the intermediate results and accumulation logic consists of adding up the results of the stages. Final-output and validity-related logic is on the right side of the schematic, meaning that prediction completion is carried out via a collection of sequential data path, rather than via an entirely combinational network.

Repeat of the wide buses, arithmetic structures flone the fact that the design is translated into realizable RTL network. The very large net count however is also a pointer that direct instantiating of all model coefficients as explicit top-level signals has the potential to cause routing congestion and synthesis overhead. Weights stored in BRAM or ROM and time-multiplexing the multiply-accumulate hardware would dramatically decrease routing complexity.

7.3 Finite-State-Machine Synthesis Result

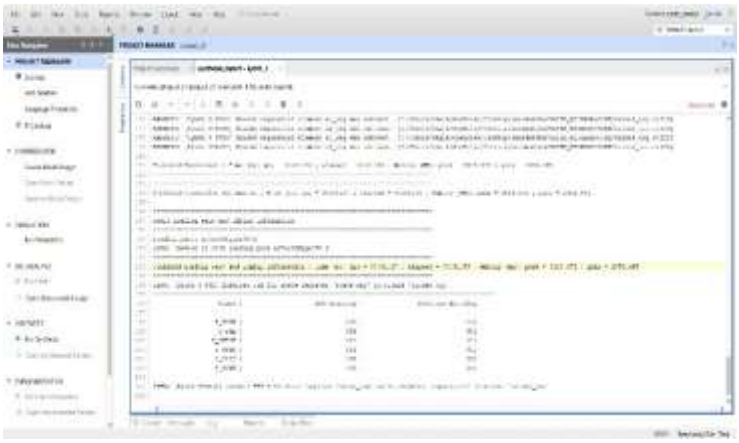

Fig.5.Synthesis log showing the inferred TriNet control FSM.

Vivado was able to locate a finite-state machine within the trinet_top module called state_reg. Six sequentially encoded states: S_IDLE, S_PRE, S_CHECK, S_PRED, S_POST, and S_DONE are present in the controller. This order corresponds to the planned prediction pipeline: receiving a request, doing preprocessing of input specifications, verifying the input validity, predicting, post-processing of the estimated circuit parameters, and making claims.

The synthesis log also documents the existence of unused sequential elements being eliminated like xi_reg, xa_reg, s1reg and s2reg. The meaning of these warnings is that the corresponding registers were not used to influence anything that could be seen in the synthesized configuration. Even though this optimization is legal, every deleted register must be considered in order to verify that no intended intermediate-learning operation has been removed, or that it has been reduced to a constant by the existing parameter values.

7.4 Synthesized Device and Power Results

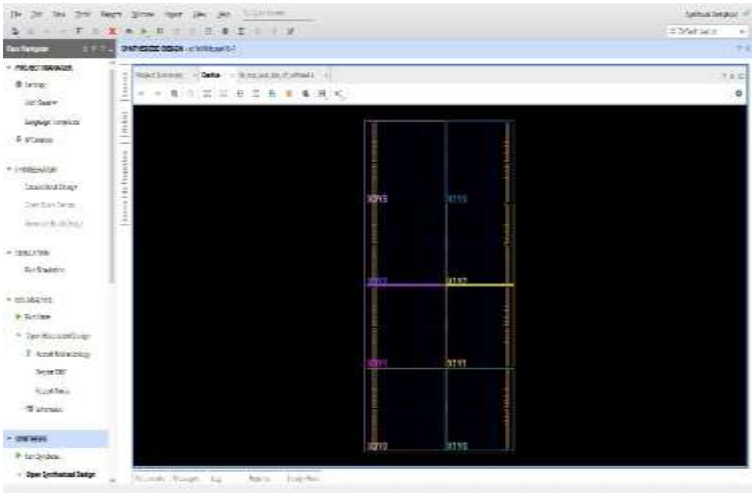

Fig.6.Synthesized device view for the selected FPGA.

The device under attack is Xilinx Artix-7 xc7s100fgga676-2. The device window displays the clock areas X0Y0 to X1Y3 which proves the fact that Vivado loaded the selected FPGA and produced a synthesized checkpoint. There is no obvious logic placement depicted in the fabric since this is an image of a synthesis-stage and, in the

project under consideration, the chosen top-level checkpoint is apparently a testbench oriented checkpoint and not a final implemented TriNet core.

7.5 Power Report

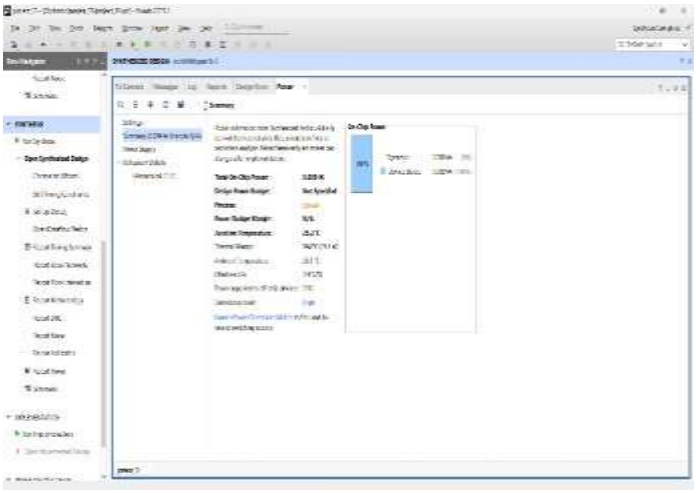

Fig.7.Post-synthesis power-estimation summary.

The power report indicates a total power on-chip of 0.089 W; dynamic power is 0.000 W and the device-static power is reported as the total 0.089 W. The junction temperature is estimated to be 25.2 C, the ambient temperature is 25.0 C and the thermal margin is 59.8 C. These values reflect the unchanging properties of the chosen FPGA with the prevailing developed netlist.

The zero dynamic-power achievement is not a realistic end of life estimate of a working TriNet accelerator. A network of normalization arithmetic, stages of multiple learning processes, accumulator, registers and control logic should consume dynamic power during clocking. The commonest cause would be that there is nothing active that can be observed as hardware in the synthesis top or the retained netlist, or propagation of switching activity was not sent into the power analysis. Hence, 0.089 W can only be used as an initial device-static estimate.

8 ADAVANTAGES, LIMITATIONS, and APPLICATIONS

The suggested system lessens the repetition of optimization in inference, gradually models intricate nonlinear associations, sieves infeasible requests, and approves of a reusable hardware deployment. It is capable of bringing about rapid initial sizing of operational amplifiers, differential amplifiers, low-power sensor interfaces, biomedical front ends, RF building blocks, automotive analog modules and educational design tools. The predictor can be turned into an intelligent design assistant by being integrated with Cadence, SPICE-based workflows, or custom EDA environments.

The main drawbacks include reliance on the clarity and faithfulness of the simulation data, topology-specific training, decreased trust beyond the design space that is calibrated, and potential loss with fixed-point quantization. The framework is used to predict parameters without substituting the final schematic, corner, Monte Carlo, layout, parasitic and post-layout verification. It is thus to be deployed as an acceleration layer in a proven analog architecture.

9 CONCLUSION

A hardware-aware TriNet driven system of automated prediction of analog circuit parameters has been introduced. The structure integrates normalized specification input, capability of learning (on three levels) and accumulation of output as well as postprocessing. The structure allows the multi-step representation of complex analog relationships by breaking down the inverse-design task into a coarse prediction followed by refinement with successive residual corrections to the prediction than a single-stage predictor. Its modular formulation (fixed points) makes it possible to implement in Verilog and possibly combine it with EDA tools. The ultimate assertions of accuracy of prediction, area, timing and strength should be determined based on the outcomes of the experiment conducted by the authors that could be directly placed in the designated Results and Discussion contents.

REFERENCES

[1] G. G. E. Gielen and R. A. Rutenbar, “Computer-aided design of analog and mixed-signal integrated circuits,” Proceedings of the IEEE, vol. 88, no. 12, pp. 1825–1852, 2000.
[2] K. S. Kundert, The Designer’s Guide to SPICE and Spectre. Boston, MA, USA: Kluwer Academic, 1995.
[3] P. E. Allen and D. R. Holberg, CMOS Analog Circuit Design, 2nd ed. Oxford, U.K.: Oxford University Press, 2002.

[4] S. Boyd and L. Vandenberghe, Convex Optimization. Cambridge, U.K.: Cambridge University Press, 2004.
[5] M. Hershenson, S. Boyd, and T. H. Lee, "Optimal design of a CMOS op-amp via geometric programming," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 20, no. 1, pp. 1–21, 2001.
[6] R. Harjani, R. A. Rutenbar, and L. R. Carley, "OASYS: A framework for analog circuit synthesis," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 8, no. 12, pp. 1247–1266, 1989.
[7] J. R. Koza, F. H. Bennett III, D. Andre, and M. A. Keane, "Automated synthesis of analog electrical circuits by means of genetic programming," IEEE Transactions on Evolutionary Computation, vol. 1, no. 2, pp. 109–128, 1997.
[8] J. Kennedy and R. Eberhart, "Particle swarm optimization," in Proceedings of the IEEE International Conference on Neural Networks, 1995, pp. 1942–1948.
[9] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," Journal of Global Optimization, vol. 11, pp. 341–359, 1997.
[10] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs. Berlin, Germany: Springer, 1996.