Automated Synthesis for Analog Computing Systems

Afolabi Ige

***Abstract*—This project proposes a Floating Gate (FG) architecture-aware synthesis tool that generates GDSII layout from analog standard cells and a netlist description. The outcomes of the work will be an open source tool that enables rapid creation of Application Specific Integrated Chips (ASIC) layout, a PDK conversion utility and FG standard cells across multiple process nodes. This work will target the conferences TCAD and DAC for publishing the results.**

I. The Argument for Analog Synthesis

Digital circuits have historically benefited from standard cell abstraction and the design automation it enables. Traditional analog circuit designs have avoided adopting this technique due daunting task of optimizing a myriad of design variables with application specific tradeoffs. However, the use of floating gate technology allows for post fabrication tuning of said variables which in turn allows for abstraction similar to that of a typical digital flow. The field of analog IC automation is currently coinciding with the explosion of alternate computing methods especially in the Machine Learning (ML) age. Adoption of such automation techniques for analog neuromorphic circuits would introduce the same advantages to allow designers to keep up with the ever-growing demand for alternate methods of computing. Floating gates crossbar structures implementing vector matrix multipliers were one of the earliest [1] references to the term "in-memory computing" and have since been shown as a viable alternative for energy efficient computing [2], [3].

Other analog synthesis tools [4], [5] attempt to procedurally generate specific circuit layout blocks as part of a larger system. The proposed technique for this current approach (fig. 1) is to use the floating gate building blocks as primitives to design a system with primarily analog elements performing the computation. Digital synthesis tools [6] are rigidly designed to expect items such as two level supplies, unidirectional signals, and clock trees instead of FG infrastructure for hot electron injection and Fowler-Nordheim Tunneling. For these reasons this project serves to fill the gap in the available open source tool chains.

Outcomes of this work:

1) 3 Modules which make up the Layout Generator Tool
2) PDK conversion utility
3) Publication submissions to DAC Conference, and TCAD Journal

Part two of this application discusses building the tool and the milestones associated with the proposed work. It also describes integration of the tool with other open source EDA tools E.g. FPAA Toolchain and Triton Route tool. Part three concerns the synthesis of applications and evaluation of performance in comparison to hand synthesized layouts as benchmarks such as a reconfigurable recurrent neural network for associative memory while also synthesizing new applications such as a low power spike sorting array.

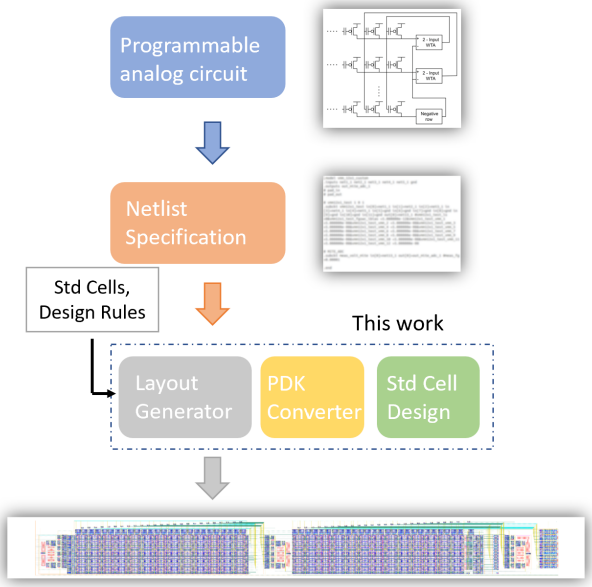

Fig. 1. The flow shows the process by which a programmable analog circuit would be synthesized down to transistor level layout. The deliverables of this project will be the layout generating tool, the pdk conversion utility and contributions to an open source analog standard cell library

II. Tool Building

A. Understanding Island Architecture

To discuss how the tool creates a system, it is important to understand "Island Architecture" as seen in fig. 2. This refers to the logical grouping of similarly functioning standard cells which are designed to tile next to each other. These analog circuits are designed to fit within a process specific pitch which can be placed into the rows of partitioned die area. This is of importance particularly because all the floating gate specific items would be placed around each island. This allows the island block to be treated as a black box with ports that can be routed to pins by existing open source tools.

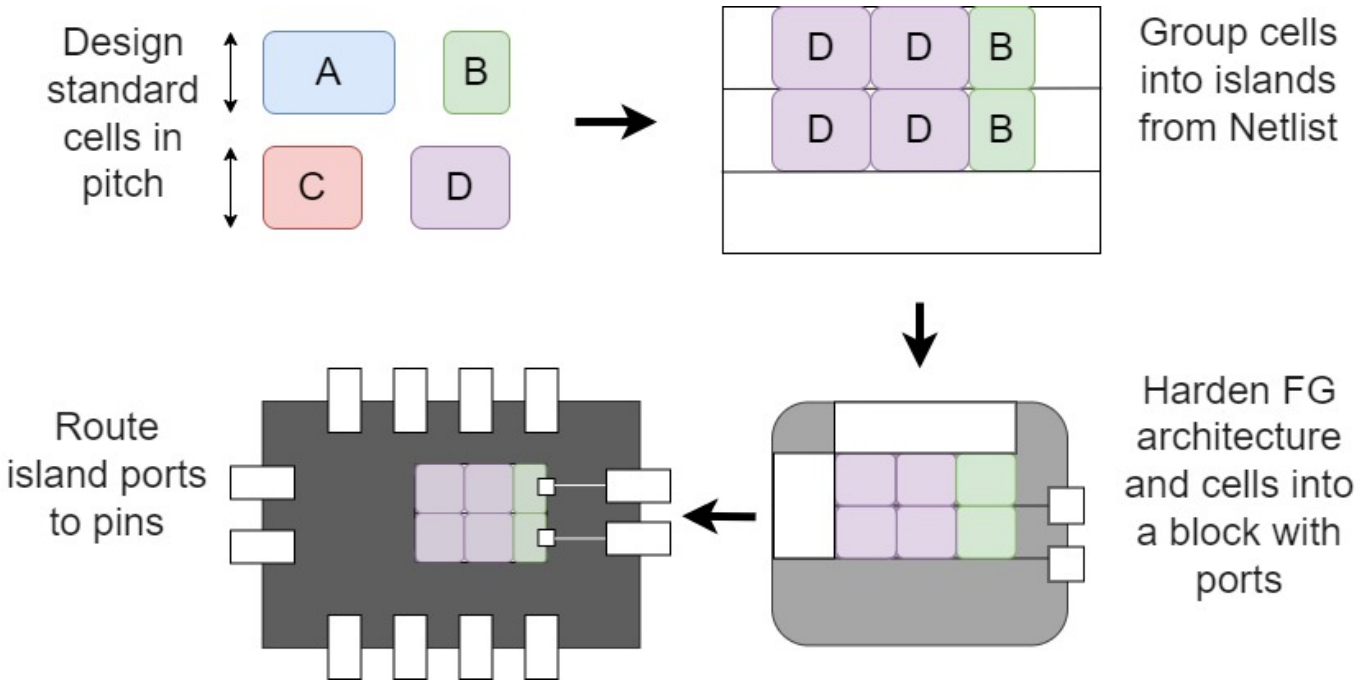

Fig. 2. Island Architecture concept highlighting the makeup of individual islands and their corresponding FG architecture.

B. Synthesis Flow

The Python-based tool is to be designed with a modular, extensible approach in mind. There are 4 modules to discuss:

1) Standard Cell Conversion Utility
2) Netlist to System Parser
3) System to Block placement
4) FG Architecture Implicit from Block placement

The flow of synthesis will follow the order listed above. The conversion utility would transfer a given layout file between process nodes. This is helpful to convert standard cells quickly between processes if need be. It will manipulate the *.gds* file to convert corresponding layers between provided layer map files. New PDKs can be added by specifying the Front End of Line (FEOL) and Back end of Line (BEOL) layers in a structured JSON format.

The Netlist-to-System parser will iterate over the provided netlist to construct system blocks. Its job is to differentiate islands from one another and represent the location of the blocks in a 2-Dimensional array to keep track of relative cell location. An object-oriented programming approach is preferred here as the cells are simply instances of the core standard cell library. These object instances will keep track of properties like port location, metals layers used etc for further analysis during evaluation.

Next, System-to-Block module will take advantage of this internal representation to convert the 2D array of cells into physical locations of the polygons to be placed when the final *.gds* is generated. A possible extensible feature for this module would be to run a design rule check after all blocks have been placed and report the status to the user.

Finally, based on the blocks sitting at the edge of the system, gate/drain side switches and decoders should be placed around the edges to complete the island structure. The final size will be constrained by the area specified by the user. If the block exceeds the dimensions, the tool will ask the user to resize their chosen island size E.g. reduce number of channels, shrink the vector matrix multiply etc.

C. Timeline

PHASE 1 LATE AUGUST	PHASE 2 SEPTEMBER - NOVEMBER	PHASE 3 JANUARY - MARCH	PHASE 4 APRIL - MAY
Std Cell Converter	Netlist Parser and Block PnR	Open Source Integration	Buffer for Review
	FG Architecture PnR	Synthesis and Benchmarking	
	TCAD and DAC Conference	CRNCH Summit Poster	

Fig. 3. Estimated timeline for individual module creation and opportunities for discussion of results. Each phase time frame also includes unit and integration test writing period.

The timeline in fig. 3 shows the allotted space given to the creation of each of the modules. Additionally, it highlights conferences that are being targeted for publication. Specifically the Design Automation Conference (DAC) deadline is mid-November while the Transaction on Computer-Aided Design Journal accepts submissions year-round on a monthly basis.

D. Open Source Integration

A detriment of open source tools in comparison to commercial counterparts is ensuring that they are able to integrate

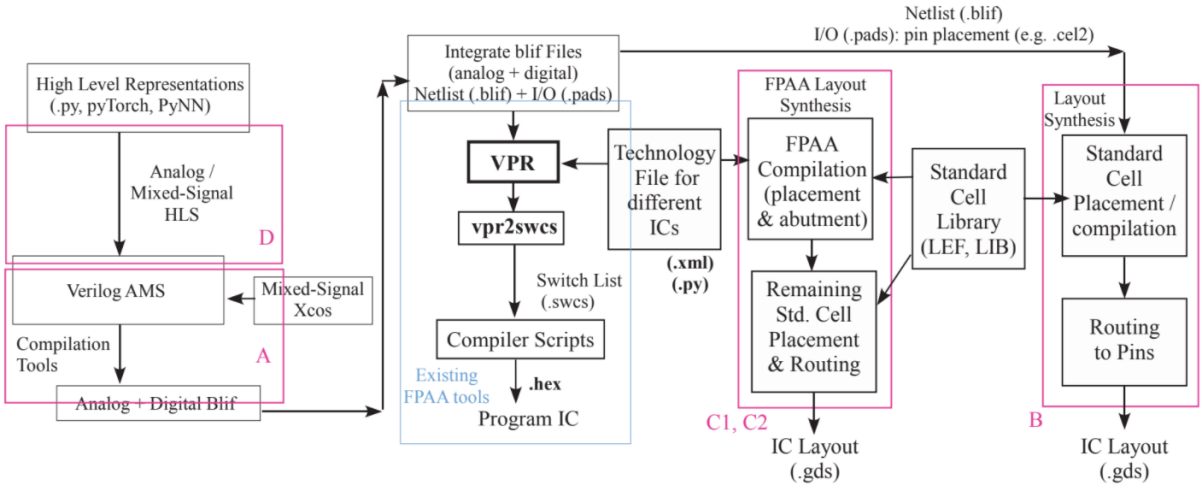

Fig. 4. Larger Flow for end-to-end Synthesis. The proposed tool would fit into the flow as described by task B above.

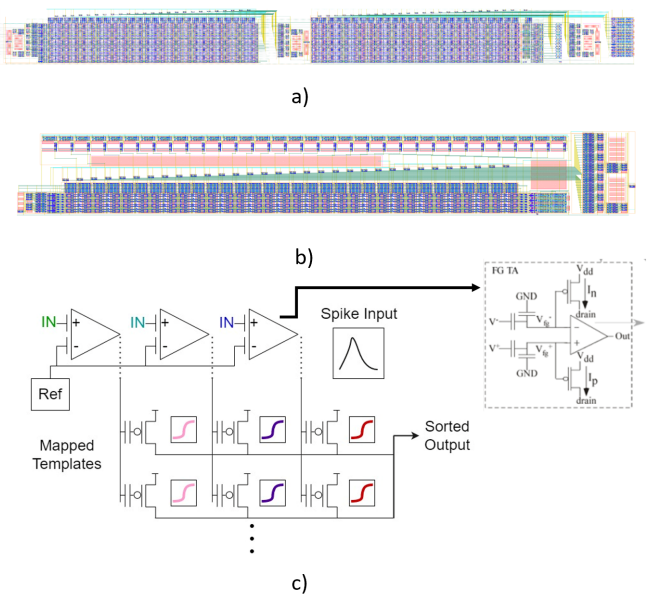

Fig. 5. (a) Shows the custom synthesized layout of a Hopfield network. This can be programmed as an associative memory that stores data in the network (b) Shows custom synthesized layout of an Arbitrary Waveform Generator. This generates custom waveforms by combining stored values in the non-volatile storage elements. (c) Simple spike sorting architecture using floating gate amplifiers feeding a vector matrix multiplier with programmed templates at each row to sort real time neural spike inputs

with other software to form a symbiotic ecosystem. This tool is designed with two major integration points in mind. The first being the higher level interface to the netlist generator tool and the second being the output provided to the pin router for detailed frame routing. As seen in fig. 4, The tool chain for Field Programmable Analog Arrays (FPAA) share a high level description with this tool. This means that the same language used for rapid prototyping can be used for specification of custom ICs. Above that tool chain is the high level programming language that can be used to target our custom hardware for ML models. This side of the flow is in collaboration with Dr Hao's Sharc Lab. The second point of integration would be connecting the tool to TritonRoute [7] to handle routing to pins on a given frame. TritonRoute was selected for its established role as part of the OpenRoad synthesis flow [6].

III. Synthesis and Evaluation

Once the tool has been developed, it is important to be able to assess its performance. To do so we will compare it to custom synthesized layouts that have been taped out. It will be compared on the following metrics:

1) Footprint (mm^2)
2) Placement Utilization (%)
3) Metal Layer Usage
4) Power Estimation

The custom synthesized layouts that will be compared against are seen in fig. 5 (a) and (b). (a) is a Hopfield network which is a fully connected recurrent neural network that sets up the output of each node and the weights between them as a dynamical system. (b) is an Arbitrary Waveform Generator which produces custom waveforms by combining stored values in the non-volatile storage elements. Transmission gates controlled by a scan chain then allow for readout of these values in quick succession. Reproducing these systems and comparing them based on the metrics listed above will demonstrate the quality of the tool.

Finally, the tool should be able to generate new systems based on a provided description. To evaluate this, a spike sorting architecture is proposed in (c). It uses a FG Operational Transconductance Amplifier (OTA) for controlled gain of

neural input signals. These are then sorted by stored templates in the VMM. The ability to create this system will show the usefulness of the tool in generating new systems. For comparison of its synthesis ability, other tools will be made to attempt generation of the same systems. Tools for comparison will be Align [5] and OpenLane [6].

IV. Conclusion

Automation of analog system synthesis would allow for quicker design cycles and much larger FG system projects executed by a few designers. Building all modules of this tool and connecting them to existing open source tools will not only accomplish the synthesis goals as outlined but will provide the opportunity for papers based on its results and contribute back to the open source community at large.

References

[1] M. Kucic, P. Hasler, J. Dugger, and D. Anderson, "Programmable and adaptive analog filters using arrays of floating-gate circuits," in *Proceedings 2001 Conference on Advanced Research in VLSI. ARVLSI 2001*, 2001, pp. 148–162.

[2] R. Chawla, A. Bandyopadhyay, V. Srinivasan, and P. Hasler, "A 531 nw/mhz, 128/spl times/32 current-mode programmable analog vector-matrix multiplier with over two decades of linearity," in *Proceedings of the IEEE 2004 Custom Integrated Circuits Conference (IEEE Cat. No.04CH37571)*, 2004, pp. 651–654.

[3] S. George, J. Hasler, S. Koziol, S. Nease, and S. Ramakrishnan, "Low power dendritic computation for wordspotting," *Journal of Low Power Electronics and Applications*, vol. 3, no. 2, pp. 73–98, May 2013. [Online]. Available: https://doi.org/10.3390/jlpea3020073

[4] H. Chen, M. Liu, B. Xu, K. Zhu, X. Tang, S. Li, Y. Lin, N. Sun, and D. Z. Pan, "Magical: An open- source fully automated analog ic layout system from netlist to gdsii," vol. 38, no. 2, 2021, pp. 19–26.

[5] K. Kunal, M. Madhusudan, A. K. Sharma, W. Xu, S. M. Burns, R. Harjani, J. Hu, D. A. Kirkpatrick, and S. S. Sapatnekar, "Invited: Align – open-source analog layout automation from the ground up," in *2019 56th ACM/IEEE Design Automation Conference (DAC)*, 2019, pp. 1–4.

[6] M. Shalan and T. Edwards, "Building openlane: A 130nm openroad-based tapeout- proven flow : Invited paper," in *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2020, pp. 1–6.

[7] A. B. Kahng, L. Wang, and B. Xu, "Tritonroute: An initial detailed router for advanced vlsi technologies," in *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2018, pp. 1–8.