# Exploring the use of 3-input Gates in Field-Coupled Nanotechnologies

Ícaro G. S. Moreira[1], Maria D. Vieira[1], Omar P. Vilela Neto[2], Ricardo S. Ferreira[1], José A. M. Nacif[1]
[1]Universidade Federal de Viçosa, Minas Gerais, Brasil
[2]Universidade Federal de Minas Gerais, Minas Gerais, Brasil
{icaro.moreira, maria.dalila, ricardo, jnacif}@ufv.br

***Abstract*—The 3-input gates have been interesting logical components for synthesis due to the circuit minimization. Moreover, the emerging Field-Coupled Nanotechnologies (FCNs) easily handle these circuits. There are 256 possible configurations for the entry set of 3-input designs, which we can group in 14 Negation-Permutation-Negation (NPN). In this scenario, AND, ORAND, and Majority are spare gates for circuit design. In this paper, we exploit the occurrence of these gates in real circuits. Our results indicate that the logical ORAND is the most adopted one due to its better coverage. This gate also can synthesize other NPN classes with fewer gates than the widely-adopted AND.**

## I. Introduction

Heuristic approaches are widely adopted to solve the complex and challenging computational task that is circuit optimization [1]. In this scenario, replacing 2-input for 3-input gates has become an additional strategy for area reduction, which leads to improvement in energy costs and heat dissipation. Moreover, the emerging Field-Coupled Nanotechnologies (FCNs) [2] have become an alternative due to their prominent simulated implementations of 3-input circuits [3]–[5], ensuring the possibility of future physical designs. The most important FCNs technologies are: Nanomagnetic Logic (NML) [6]–[9], Quantum-dot Cellular Automata (QCA) [10], [11], and Atomic Silicon Quantum Dot (SQD) [3], [12]–[14].

Marakkalage *et al.* [1] investigate the trade-offs in terms of energy and area for the 3-input gates. In more detail, this work exploits the 256 possibilities of outputs for them. It is possible to group these configurations in 14 Negation-Permutation-Negation (NPN), where two classes use, at most, two inputs. Hence, there are ten NPN gates to classify all existing 3-input configurations. In this scenario, the NPN classification is a well-solved problem. The related works explore heuristic methods such as hierarchical classifications [15], function polarization analysis [16], AIG structures [17], [18], analyzing variables symmetry [19], one pre-computed library created from LUT's topology analysis [20].

Marakkalage *et al.* [1] show the 3-input NPN classes, where some of them can minimize the average number of gates to produce the other NPN classes. Consequently, the use of these designs results in improvements in area and energy costs. In this paper, we explore the occurrence of these gates in real circuits. First, we use the ABC [21] to obtain the hardware descriptions for circuits composed of gates with at most 3 inputs for the well-known ISCAS89 [22] benchmarks. Next, we use this software to obtain the truth tables for all Boolean expressions from these given designs. Then, we use our hardware-based NPN classifier to match these truth tables with 3-input NPN classes. Finally, we produce a detailed summary containing the number of gates from each NPN class that compose the benchmarks. Our main contributions are: (1) the gate occurrence summary and (2) a hardware-based NPN classifier for 3-input gates.

We organize this paper as follows. In Section II we describe prior approaches for similar combinatorial problems. In Section III, we depict the theoretical basis of the NPN classes and also show an SQD-based 3-input gate. In Section IV, we present our approach to summarize gate occurrence and also propose our hardware accelerator approach for NPN classifying. Section V shows the gate occurrence summary, the number of gates from each NPN class to generate the ISCAS89 [22] benchmarks, and compares the execution times between our NPN classifier and prior works. In Section V we conclude and present our future work.

## II. Related Work

This section presents widely-known solutions for our target problem once the NPN classification is a well-solved problem. Z. Huang *et al.* [16] use a heuristic based on a three-step algorithm for a 6-16 input Boolean function, where the output depends only on the number of true inputs. On the other hand, Benini *et al.* [23] present a BDD-based approach. Another widely-adopted approach is the class hierarchy to pre-classify utilizing a heuristic approach saving intermediate results (i.e., the truth tables) [15]. This early step reduces the execution times due to the possibility of sopping the execution when the function becomes equal to a hierarchy representative.

Soeken *et al.* [18] propose an algorithm for NPN classification using AIGs (and inverter graph) and LEXSAT (a variant of SAT problem). This approach uses AIGs composed of 2-input AND gates to represent the Boolean functions, which are compared by the LEXSAT. The AIG structures also could classify Boolean functions for a solution based on the Galois graph for the topology representation [24]. The pre-computed libraries of implementable NPN equivalent functions could support this purpose [20].

## III. Background

This section describes NPN classes and their classifications. It also discusses the importance of using them in discovering new gates, and why it is crucial to classify one gate for circuit design. The NPN class consists of a group of logic gates that can be obtained from others negating or permuting the inputs of an initial gate or negating the output. We cover two different methods of performing the classification of these gates. Each class has a unique configuration, which also depends on the size of gate inputs. There are $2^{2^n}$ possibilities of output for a gate with size n for the entry set.

TABLE I
NPN CLASSES FOR 3-INPUT GATES [1]

| Gate | Logical Expression |
|---|---|
| And3 | $x \wedge y \wedge z$ |
| XorAnd | $x \wedge (y \oplus z)$ |
| OrAnd | $x \wedge (y \vee z)$ |
| Onehot | $x\neg y\neg z \oplus \neg xy\neg z \oplus \neg x\neg yz$ |
| Majority | $< xyz >$ |
| Gamble | $x \wedge y \wedge z \oplus \neg x \wedge \neg y \wedge \neg z$ |
| Dot | $x \oplus (z \vee x \wedge y)$ |
| Mux | $x?y : z$ |
| AndXor | $x \oplus y \wedge z$ |
| Xor3 | $x \oplus y \oplus z$ |

Marakkalage *et al.* [1] exploit the 3-input NPN classes. Each gate group has 256 possible output configurations. It means 256 boolean functions that could be grouped in 14 NPN classes, where four of them need at most two inputs, resulting in ten 3-input NPN classes, shown in Table I. Figure 1 shows designs from a same 3-input NPN class. In other words, Figure 1(a), Figure 1(b), Figure 1(c), and Figure 1(d) are NPN equivalent, which means could be obtained from each other through complements and permutations. In this context, Figure 1 shows the original gate (a) and examples of modified versions from the same NPN-classes that we can obtain from 3 kinds of changes: (b) inputs permutation, (c) the use of input complement, and (d) the use of output complement.

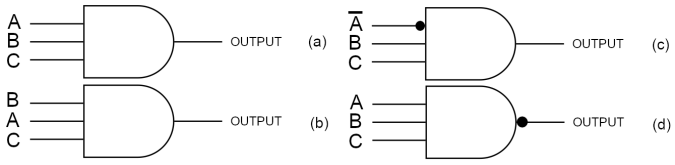


Fig. 1. Configurations from a same NPN class. (a) Original AND gate. (b) inputs permutation. (c) input A complement. (d) output complement.

In this context, Table II shows the 3-input gate count for synthesize the NPN classes with 3, 4, and 5 inputs [1]. These values show that the designs with fewer gates can be produced with the gates from the ORAND class, considering any of these input counts. Therefore, 3-input designs from this NPN class are standout gates for circuit reduction, once it could reduce the gate count, and consequently both area and energy.

As above-mentioned, the FCNs have prominent implementations of 3-input gates [4], [25].

TABLE II
3-INPUT GATES COUNT TO SYNTHESIZE ALL X-INPUT NPN CLASSES [1]

| X | AND | Majority | ORAND |
|---|---|---|---|
| 3 | 33 | 30 | 25 |
| 4 | 1134 | 1036 | 883 |
| 5 | 1664 | 1612 | 1302 |

These technologies overcome the physical limits of the current CMOS, allowing the production of high-performance, low-power computing devices. Furthermore, FCNs could also integrate circuits as extensions of existing CMOS designs. FCN uses nanomagnetic interactions (NML) or Coulombic interactions between electrons (QCA and SQD) to represent the logic levels.

The emerging SQD has become a relevant alternative in this scenario due to the sturdy physical synthesis and ease of 3-input gates simulation. Vieira *et al.* [3] present the novel SQD-based ANDOR and ORAND gates. Both are correspondent to a NPN class. Figure 2 depicts these designs that are composed of the emerging Silicon-Dangling-Bound. This structure originates from the desorption of single hydrogen atoms on a Silicon surface.

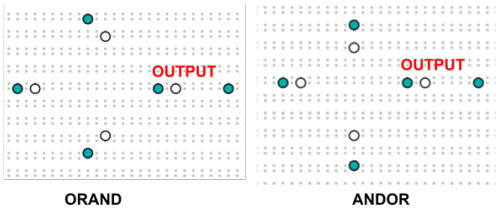


Fig. 2. SQD-based 3-input gates. (a) ORAND. (b) ANDOR.

## IV. Methodology

This section depicts our main contributions. Section IV-A presents our workflow to produce the gate occurrence summary. Section IV-B shows our hardware-based approach to classify 3-input gates into the main NPN classes. We focus on the report on the importance of some spare 3-input gates to synthesize real circuits. Therefore, we only produce an NPN classifier as an auxiliary tool with a reasonable execution time, which does not need to be competitive.

### A. Our workflow: Gate occurrence analysis

The choice of gates to compose circuits has a significant impact on circuit minimization. Following this statement, this work exploits the 3-input gates to find one that can reduce the total area cost. We evaluate three widely-adopted gates for circuit design: AND, Majority, and ORAND. Another advantage of this choice consists in the possibility to implement these gates in FCNs (Field Coupled Nanotechnologies) [4] such as SQD (Silicon Quantum Dot) and QCA (Quantum-dot Cellular Automata). We use the ISCAS89 benchmarks [22] as an example of real circuits.

Figure 3 depicts the workflow for gate analysis, starting from the bench file to the exact classification of the NPN class.

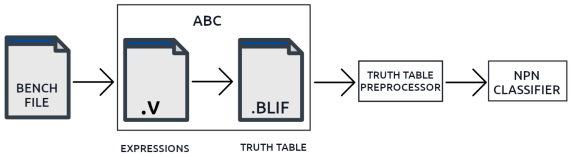


Fig. 3. Workflow: gate occurrence analysis.

We use the well-established tool for sequential synthesis and verification, named ABC [21], to generate circuits composed by gates with at most three inputs. First, we read the bench file which contains one hardware description from the ISCAS89 benchmarks [22], using the command **read_verilog benchfile.v**. Then, we execute the command **resyn2** which synthesizes using combinational logic, generating a new Verilog file. Next, we use the command **if -K 3** to decompose the circuit into a 3-LUT, in other words, it transforms the circuit into 3-input functions. To finish the ABC usage step, we execute the command **write_blif filename.blif** for create a new .blif file that contains the transformed circuit.

The last two steps are our main contributions: classifying gates into NPN classes and produce the gate occurrence summary. The first one is a high-level-based algorithm to extract truth tables from the blif file given by ABC. In this step, we do not consider the truth tables with 1-input (inverters) and the 2-input. Therefore, we only classify in NPN classes the remaining designs with three literal inputs, thus not considering gates with constant inputs. Finally, we match these 3-input truth tables with the NPN classes. Then, we could report the incidence for each class in benchmarks.

### B. Our hardware-based NPN classifier

We develop a hardware-based approach to classify gates into NPN classes. In a preprocessing step, we define one logical expression to represent each gate group, according to Figure I. Then, our algorithm obtains all combinations of inputs and outputs for each NPN class, using permutations and the addition of inverters. Since we receive a logical entry statement at run-time, we must match it with an NPN class. Notice that permuting entries or adding inverters in a prior gate produces changes in the expected output. For instance, Figure 4 depicts the permutation of X to Y for a 2-input configuration. This operation induces the change between the second and third lines from the truth table, generating a new output.

## V. Results

We organize this section as follows. Section V-A presents our summary for 3-input gate occurrence, evaluating the coverage of AND, ORAND, and Majority. Section V-B shows our analysis above the related work.

### A. Summary: gate occurrence (And, Orand, Majority)

Table III shows the gate occurrence for all 3-input NPN classes, using circuits from well-known benchmarks ISCAS89 [22]. It is important to emphasize that gates such

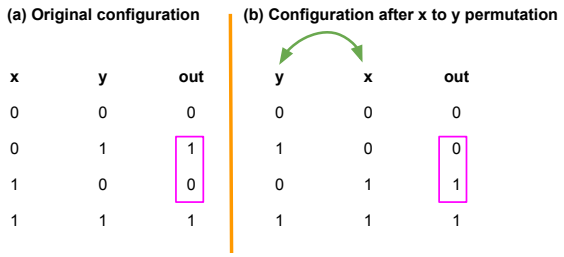


Fig. 4. High-level code: X to Y permutation

TABLE III
Gate occurrence

| Gates | s838 | s1196 | s1423 | s5378 | s9234 | Total |
|---|---|---|---|---|---|---|
| **And** | **46** | **69** | **26** | **117** | **111** | **369** |
| Xorand | 0 | 3 | 43 | 12 | 52 | 110 |
| **Orand** | **33** | **109** | **63** | **176** | **219** | **600** |
| Onehot | 0 | 0 | 0 | 0 | 0 | 0 |
| **Majority** | **0** | **0** | **0** | **0** | **0** | **0** |
| Gamble | 0 | 0 | 0 | 0 | 0 | 0 |
| Dot | 0 | 2 | 0 | 0 | 0 | 2 |
| Mux | 1 | 9 | 23 | 4 | 69 | 106 |
| Andxor | 11 | 0 | 1 | 8 | 9 | 29 |
| Xor | 0 | 0 | 0 | 28 | 8 | 36 |

| Gates | s13207 | s15850 | s35932 | s38417 | s38584 | Total |
|---|---|---|---|---|---|---|
| **And** | **192** | **227** | **16** | **481** | **1048** | **1964** |
| Xorand | 87 | 103 | 544 | 209 | 180 | 1123 |
| **Orand** | **227** | **285** | **1120** | **942** | **1171** | **3745** |
| Onehot | 0 | 1 | 0 | 0 | 0 | 1 |
| **Majority** | **0** | **0** | **0** | **0** | **0** | **0** |
| Gamble | 0 | 1 | 0 | 12 | 0 | 13 |
| Dot | 7 | 5 | 0 | 10 | 85 | 107 |
| Mux | 139 | 247 | 16 | 961 | 1144 | 2507 |
| Andxor | 18 | 22 | 16 | 37 | 92 | 185 |
| Xor | 5 | 14 | 288 | 16 | 6 | 329 |

as XOR, which are widely used in adder circuits, have a significant presence in the analyzed circuits for this reason. Otherwise, gates like Majority, Onehot, and Gamble are considerably rare. Marakkalage *et al.* [1] show that the ORAND NPN Class appears more frequently than the Majority and And3 NPN Classes. Our results complement this information, proving that this NPN group is the most frequent in these benchmarks. Furthermore, we notice that the AND is in the top 3 most frequent for several benchmarks, as expected as the synthesis algorithm targets AIGs.

Table IV presents the total number of 3-input gates, considering that we only use one NPN class to compose the ISCAS89 [22] circuits. Marakkalage *et al.* [1] present the number of gates from each NPN class to synthesize the other classes. Therefore, we only multiply these numbers from the related work by the gate count per benchmark given by Table III. The Majority gate frequently appears as the NPN class that uses fewer gates to compose the benchmarks, followed by ORAND, at second position.

### B. Comparisons with Related Work

The NPN classification is already a well-explored problem in the literature. Therefore, there are many widely adopted

TABLE IV
3-INPUT AND, ORAND, AND MAJORITY: GATE COUNT FOR ISCAS89

| **Bench** | **s838** | **s1196** | **s1423** | **s5378** | **s9234** |
|---|---|---|---|---|---|
| AND | 252 | 586 | 532 | 1119 | 1531 |
| ORAND | 182 | 388 | 398 | 770 | 1056 |
| Majority | 182 | 381 | 269 | 678 | 884 |
| **Bench** | **s13207** | **s15850** | **s35932** | **s38417** | **s38584** |
| AND | 2223 | 3059 | 7392 | 9138 | 12251 |
| ORAND | 1527 | 2041 | 5664 | 5788 | 7739 |
| Majority | 1263 | 1708 | 3456 | 5139 | 7272 |

methods. Even so, our approach achieves a satisfactory average result regarding our need for only 3-input gates and ignoring scalability.

TABLE V
EXECUTION TIMES IN MILLISECONDS FOR NPN-CLASSIFIERS

| | [20] | [16] | [15] | [18] | [19] | our |
|---|---|---|---|---|---|---|
| Time | 300 | 5180 | 2690 | 5650 | 60 | 6500 |
| Inputs | 3 | 6 | 6 | 6 | 6 | 3 |

## VI. CONCLUSIONS AND FUTURE WORK

We exploit the occurrence of gates from all 3-input NPN classes in real circuits, using the well-known benchmarks ISCAS89 [22]. Our results show that the ORAND is the most incident gate for these benchmarks. Furthermore, we could produce these real circuits only using the ORAND gate with costs that are competitive with the widely-adopted AND. Furthermore, the related work shows that the ORAND also compose NPN classes with 3, 4, 5, and 6 inputs with fewer gates than AND and Majority. Also, a prior work presents an FCN-based implementation of the logical ORAND using the emerging SQD technology.

We aim to adjust our NPN classifier also to match 2-input gates with the 3-input NPN classes. Furthermore, we will investigate the profitability of the most prominent 3-input gates for a specific target technology such as FCNs. We also aim to improve the execution times of our NPN classifier to exploit circuit classes with 4 or more inputs.

## ACKNOWLEDGMENTS

The authors thank UFV, and the agencies CAPES, CNPq, and FAPEMIG for partially funding this work.

## REFERENCES

[1] D. S. Marakkalage, E. Testa, H. Riener, A. Mishchenko, M. Soeken, and G. De Micheli, "Three-input gates for logic synthesis," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 2020.

[2] R. E. Formigoni, R. S. Ferreira, and J. A. M. Nacif, "A survey on placement and routing for field-coupled nanocomputing," *Journal of Integrated Circuits and Systems*, vol. 16, no. 1, pp. 1–9, 2021.

[3] M. D. Vieira, Ícaro G. S. Moreira, P. A. R. L. Silva, L. O. Luz, R. S. Ferreira, O. P. V. Neto, and J. A. M. Nacif, "Novel three-input gates for silicon quantum dot," in *2018 31st Symposium on Integrated Circuits and Systems Design (SBCCI)*, 2021.

[4] A. N. Bahar, K. A. Wahid, S. S. Ahmadpour, and M. Mosleh, "Atomic silicon quantum dot: A new designing paradigm of an atomic logic circuit," *IEEE Transactions on Nanotechnology*, vol. 19, 2020.

[5] J. Das, S. M. Alam, and S. Bhanja, "Addressing the layout constraint problem when cascading logic gates in nanomagnetic logic," in *IEEE International Conference on Nanotechnology (IEEE-NANO)*, 2012.

[6] R. E. Formigoni, O. P. V. Neto, and J. A. M. Nacif, "Bancs: Bidirectional alternating nanomagnetic clocking scheme," in *2018 31st symposium on integrated circuits and systems design (SBCCI)*. IEEE, 2018, pp. 1–6.

[7] R. E. Formigoni, R. S. Ferreira, and J. A. M. Nacif, "Ropper: A placement and routing framework for field-coupled nanotechnologies," in *2019 32nd Symposium on Integrated Circuits and Systems Design (SBCCI)*. IEEE, 2019, pp. 1–6.

[8] R. E. Formigoni, L. L. A. Vieira, O. P. V. Neto, R. Ferreira, and J. A. M. Nacif, "Evaluating nanomagnetic logic circuit layouts using different clock schemes," *Analog Integrated Circuits and Signal Processing*, vol. 106, no. 1, pp. 205–218, 2021.

[9] L. O. Luz, J. A. M. Nacif, R. S. Ferreira, and O. P. V. Neto, "Nmlib: A nanomagnetic logic standard cell library," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2021, pp. 1–5.

[10] P. A. R. Silva, J. R. S. Alves, R. S. Ferreira, O. P. V. Neto, and J. A. M. Nacif, "A novel five-input multiple-function qca threshold gate," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2018.

[11] F. S. Torres, P. A. Silva, G. Fontes, J. A. Nacif, R. S. Ferreira, O. P. V. Neto, J. Chaves, and R. Drechsler, "Exploration of the synchronization constraint in quantum-dot cellular automata," in *2018 21st Euromicro Conference on Digital System Design (DSD)*. IEEE, 2018, pp. 642–648.

[12] E. Varga, A. Orlov, M. T. Niemier, X. S. Hu, G. H. Bernstein, and W. Porod, "Experimental demonstration of fanout for nanomagnetic logic," *IEEE Transactions on Nanotechnology*, vol. 9, no. 6, 2010.

[13] F. Sill Torres, R. Wille, P. Niemann, and R. Drechsler, "An energy-aware model for the logic synthesis of quantum-dot cellular automata," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 12, pp. 3031–3041, 2018.

[14] M. B. Haider, J. L. Pitters, G. A. DiLabio, L. Livadaru, J. Y. Mutus, and R. A. Wolkow, "Controlled coupling and occupation of silicon atomic quantum dots at room temperature," *Phys. Rev. Lett.*, vol. 102, p. 046805, Jan 2009.

[15] A. Petkovska, M. Soeken, G. De Micheli, P. Ienne, and A. Mishchenko, "Fast hierarchical npn classification," in *Field Programmable Logic and Applications (FPL)*, 2016.

[16] Z. Huang, L. Wang, Y. Nasikovskiy, and A. Mishchenko, "Fast boolean matching based on npn classification," in *2013 International Conference on Field-Programmable Technology (FPT)*, 2013, pp. 310–313.

[17] V. Possani, "Parallel algorithms for scalablelogic synthesis verification." UFRGS, 2019.

[18] M. Soeken, A. Mishchenko, A. Petkovska, B. Sterin, P. Ienne, R. Brayton, and G. Micheli, "Heuristic npn classification for large functions using aigs and lexsat," in *SAT*, 2016.

[19] X. Zhou, L. Wang, P. Zhao, and A. Mishchenko, "Fast adjustable npn classification using generalized symmetries," in *International Conference on Field Programmable Logic and Applications (FPL)*, 2018.

[20] Chen, Lei, "Post-mapping topology rewriting for fpga area minimization," 2009. [Online]. Available: http://hdl.handle.net/10012/4601

[21] "Berkeley logic synthesis and verification group, abc: A system for sequential synthesis and verification," http://www.eecs.berkeley.edu/~alanmi/abc/, release: 2021-06-19.

[22] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," in *IEEE International Symposium on Circuits and Systems,*, 1989, pp. 1929–1934 vol.3.

[23] L. Benini, M. Favalli, G. Micheli, and C. Stanford, "Generalized matching : a new approach to concurrent logic optimization and library binding," 1995.

[24] H. N. Chiu, S. S. H. Ng, J. Retallick, and K. Walus, "Poissolver: a tool for modelling silicon dangling bond clocking networks," in *2020 IEEE 20th International Conference on Nanotechnology (IEEE-NANO)*, 2020.

[25] A. N. Bahar, S. Waheed, N. Hossain, and M. Asaduzzaman, "A novel 3-input xor function implementation in quantum dot-cellular automata with energy dissipation analysis," *Alexandria Engineering Journal*, vol. 57, no. 2, 2018.