DiSCERN: Distilling Standard-Cells for Emerging Reconfigurable Nanotechnologies

Shubham Rai, Michael Raitza, Siva Satyendra Sahoo, Akash Kumar
Chair For Processor Design, CfAED, Technische Universität Dresden, Dresden, Germany
<shubham.rai, michael.raitza, siva_satyendra.sahoo, akash.kumar>@tu-dresden.de

Abstract—**Recent attempts on circuits based on emerging reconfigurable nanotechnologies have primarily focused on using the traditional CMOS design flow involving similar-styled standard-cells. In the present work, we show that logic gates which implement self-dual functions can be efficiently implemented using reconfigurable nanotechnologies. We propose an algorithm which analyses the truth-tables of cuts in a mapped circuit to list all such potential reconfigurable logic gates for a particular circuit. Technology mapping with these new logic gates (or standard-cells) leads to a better mapping in terms of area and delay. Experiments employing our methodology over EPFL benchmarks, show average improvements of around 13%, 16% and 11.5% in terms of area, number of edges and delay respectively as compared to the conventional CMOS-centric standard-cell based mapping.**

I. Introduction

An interesting class of devices which show ambipolar behavior has been demonstrated based on various materials like silicon [5, 6], germanium [15], carbon [2] etc. Transistors made with such materials demonstrate tunable polarity which implies exhibiting both p and n-type behavior from a single device. Such behavior when abstracted in the form of logic gate, results in showcasing reconfigurable functionality on the application of different potential biases [20]. Various circuit-level implementations based on reconfigurable FETs (RFETs) have been shown to offer better area and delay results as compared to the conventional CMOS-based implementations [14, 16].

An integral component in development of emerging technologies, is to develop efficient design automation steps which can compliment these feature-rich devices. Earlier works like [11, 20] demonstrated efficient reconfigurable logic-gates using RFETs but were hand-designed and were derived from CMOS-like standard-cells. In the present work, we explore the reason why only certain logic gates can be implemented as reconfigurable logic gates and how such efficient reconfigurable logic gates can be designed. We show that logic gates based on self-dual functions are more favorable choice for standard-cell implementation based on RFETs. The major contributions are summarized below:

- We present a methodology to identify reconfigurable logic gates which can be efficiently implemented using reconfigurable nanotechnology and can lead to optimized circuit implementation in terms of metrics like area, power and delay.
- We propose an algorithm which can distill probable standard-cells from a circuit's logic network based on truth tables of individual cuts of a mapped circuit. These probable standard-cells are a direct outcome of the requirements of the logic implementation of individual circuits and hence can lead to better mapping in terms of area, delay and edge connections.

Using our algorithm over EPFL benchmark suite [10], we first generate a list of standard-cells. Upon adding these standard-cells to the technology independent generic library, mapping on the same EPFL benchmarks leads to improved area (13%), delay (11.5%) and edge connections (16%). Further, we compare our approach with the state-of-the art work on technology mapping for these reconfigurable nanotechnologies [17] and get improvements in both area and edge connections without any compromise on the delay. The whole flow is available online under open source license to enable further research [21]. The remainder of this paper is organized as follows. Section II gives the background of RFET devices and previous works. Section III describes about how interchangeable reconfigurability occurs in RFETs-based circuits. Section IV details about the approach to extract standard-cells. Experiments and discussions are explained in section V. Concluding remarks are given in section VI

II. Background and Motivation

A. Reconfigurable Nanotechnologies

Various nanodevices based on different materials and geometries have been proposed which exhibit tunable polarities. These devices exhibit electrical symmetry in both p- and n-type functionality. This electrical symmetry translates into functional symmetry which allows logic gates to be reconfigurable. RFETs come with two types of gate terminals– *Program Gates* (PG), which controls the direction of the flow of charge carriers, and the *Control Gate* (CG) which controls the flow of the charge carriers. For the RFET to be in the on-state, both the PG and CG should have same inputs [5].

B. Multiple Gate Terminals on the Same Channel

A prominent feature which has been demonstrated both experimentally [9] and theoretically [13] is that RFETs can have two or multiple independent gate terminals on a single channel. These multiple independent gate terminals (both pogram and multiple control gates) behave as wired-AND and only allow current when all the gates are in on-state [18]. This feature allows building larger logic gates with lesser number of transistors as compared to their CMOS implementation. Hence, gates like XOR3, MIN which were generally avoided in CMOS, due to their slower performance can be efficiently realized with RFETs [20].

C. Previous Works

Most of the previous work for enabling design flow for emerging reconfigurable nanotechnologies has been primarily focused at the logic synthesis level [12, 19]. However, the generic library used in their experiments still had the CMOS-styled logic gates. Apart from the above works, several works on designing RFET-centric circuit implementations have been shown in [8, 14, 20]. In order to support reconfigurable logic gates based on silicon nanowire based RFETs, the authors in [17] proposed higher order functions in order to use logic gates proposed in [13] to bring out gains in area during technology mapping stage. To the best of our knowledge, this was the only work focusing on technology mapping for RFETs. However, the standard-cell used were derived from CMOS-styled logic gates. In the present work, we target to

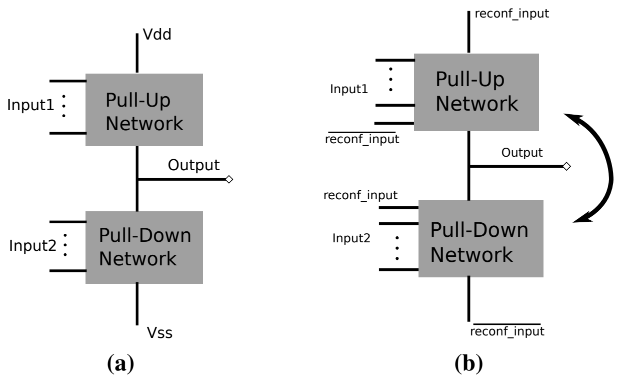


Fig. 1: (a) Fixed Pull-up and pull-down network in case of Complimentary MOS logic gates. (b) Interchangeable Pull-up and pull-down network in case of RFET-based logic gates. The reconf_input decides the logic functionality

find the methodology in order to design logic gates which can demonstrate reconfigurability.

III. Encapsulating Pull-Up and Pull-Down Reconfigurability

In the present section, we will explore the reason behind why only certain logic gates exhibit reconfigurability and not others.

A. *Interchangeable Pull-Up and Pull-Down Networks*

In conventional CMOS based logic gates, there is a distinct demarcation between the PMOS and NMOS types. CMOS-based logic gates have a *Pull-Up* network (pun) made of PMOS transistors which drives the output of the logic gate to logic 1. The inverse is the *Pull-Down* network (pdn) consisting of NMOS transistors which drives the output to logic 0. This is shown in Fig. 1a. Since PMOS devices are slower as compared to NMOS due to lesser mobility of holes as compared to the electrons, hence, designing logic gates based on CMOS often requires careful design of channel widths to achieve similar drive strength of the pull-up and pull-down networks.

In contrast, RFETs do not have such imbalance between the p and n-type behavior as they are functionally symmetric [5]. Hence, by changing the potential at program gate (and correspondingly at control gate so that the device is in the on-state), both p- and n-type behavior is realized from a single RFET. When this change of potential at the program gate is done to logic gate, the pull-up and pull-down network can be interchanged and hence more than one logical functionality can be realized. In the Fig. 1b, the potential change at $\overline{reconf\_input}$ switches the pull-up and pull-down network. In case of logic gates proposed in [20], the reconfigurable logic gates demonstrated the interchangeable pull-up and pull-down network based on a single reconfigurable input.

B. *Reconfigurable Truth-Table*

The truth-table of a logic gate represents the electrical characteristics in terms of logic 1 and logic 0. The concept of self-duality is integral in understanding reconfigurable truth-table as it correctly represents interchangeable pull-up and pull-down network in a logic gate.

Self-Dual Function A logic function $f(x_1, x_2, ..., x_n)$ is said to be self-dual if $f(x_1, x_2, ..., x_n) = \bar{f}(\bar{x_1}, \bar{x_2}, ..., \bar{x_n})$ [7]. By complementing the function, an equivalent self-dual formulation is $\bar{f}(x_1, x_2, ..., x_n) = f(\bar{x_1}, \bar{x_2}, ..., \bar{x_n})$. For a particular instance of $x_1, x_2...x_n$, $f(x_1, x_2, ..., x_n)$ and $\bar{f}(\bar{x_1}, \bar{x_2}, ..., \bar{x_n})$ are defined as a pair of *dual terms*. A *flip* on a literal is defined as replacing a variable by its comlement.

C	B	A	Output
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

$= \underbrace{1110}$

Splitted Truth Table are bitwise complementary. This implies reconfigurable pull-up and pull-down networks

$= \overbrace{0001}$

Fig. 2: Truth-Table for Minority logic gate. The truth-table is split over the value of C which is the reconfigurable input.

Fig. 2 shows an example of a self-dual function, *Minority*. Here, when the truth-table is divided over the value of C (or any other arbitrary literal per se), the two parts of the output are self-dual.

Theorem 1. *All self-dual functions can be implemented as a static reconfigurable logic gate using any single reconfiguration input.*

Proof. For the proof, we consider two domains – a Boolean domain which deals with Boolean functions and the electrical domain where a static complementary logic network implements a Boolean function. As stated earlier, reconfiguration is performed by changing the potential at the program gate of respective transistors. This means, reconfiguring a logic network changes each involved PMOS transistor (pull-up network) to NMOS transistor (pull-down network) and vice versa. Every self-dual function can be implemented as a static logic network of pull-up and pull-down networks.

In a dual term, each variable can reconfigure a maxterm into its dual minterm. Since every signal of the static logic network represents a variable of the reconfigured function, any of the signals can be used to reconfigure the circuit. However, for the networks of transistors to work, all the remaining signals need to be flipped so that the pull-up and pull-down networks get interchanged in a symmetrical way which is possible only with RFETs. While transistors cannot grow arbitrarily large, say, they can accommodate up to k signals, they have to be split into series of multiple transistors. Each is reconfigured by the same signal and connects to $k - 1$ remaining signals. □

IV. Finding Efficient Standard-Cells

In the present section, we propose an approach to extract possible standard-cells from a given circuit. We use the underlying pattern in the truth-table to find self-dual function as discussed in the previous section. Our approach is to traverse through a circuit and observe the number of specific cuts in a logic network which has the potential to be mapped to a reconfigurable logic gate which implements a self-dual function. We focus on cuts as cut-enumeration is an integral step in technology mapping [3] for both LUT-based or standard-cell-based mapping.

These reconfigurable logic gates which are extracted only make logical sense with 3 or more inputs. Two input reconfigurable logic function is a trivial function. as it represents a reconfigurable function of a single input. The CMOS-based implementation for such distilled reconfigurable logic gates are not efficient in terms of number of transistors and performance [20]. Hence, having these logic gates available in the RFET-based generic library, can potentially improve the technology mapping of circuits.

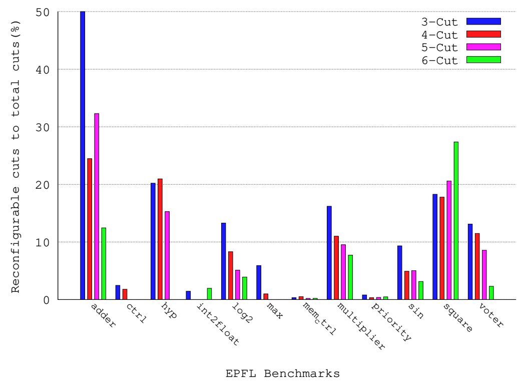


Fig. 3: Ratio of cuts with reconfigurable truth-table to total number of cuts for k-input cut based mapping where k = 3,4,5,6 respectively.

V. Experiments and Discussion

The whole experiment setup is divided into two major parts. In the first part (V-A), we populate a histogram of the reconfigurable logic gates present in the circuit. For this analysis, we use the EPFL benchmark suite [10]. We use ABC tool [4] to enumerate the varying k-input cuts for individual benchmarks. We carry out analysis for four runs with values of k as $3, 4, 5, 6$ respectively. In the second part, we use the reconfigurable logic gates found in the first part as standard-cells to constitute a generic library. We carry out a standard-cell based mapping to analyze the effect of these newly added standard-cells on the overall area, delay and edge connections of individual benchmarks.

A. Distilling Standard-Cells

Fig. 3 shows the ratio of cuts which exhibit reconfigurable truth-table to the total number of cuts in a logic network. The various colored bars show the results for cut-based mapping for different values of k. A crucial point to note here is that the number of such cuts with reconfigurable truth-tables entirely depends upon the underlying functionality of the circuit under test. We hereby explain the various empirical findings.

The first observation is that most of the benchmarks have a high percentage of such cuts with reconfigurable truth-table. Benchmarks like $adder$, $square$, $multiplier$ show a higher percentage $(20 - 50\%)$ of such cuts. A second observation is that in case of smaller values of k, the ratio is larger as compared to when k is more. For example in case of benchmarks like $adder$ and $multiplier$, this ratio somewhat decreases with increasing values of k. This is due to the fact, that with smaller value of k, the circuit will have more number of cuts. However, this is not a uniform observation. For instance, in case of $square$ benchmarks, the trend is opposite. This is possible because in this case, smaller non-reconfigurable cuts got subsumed in larger cuts which show reconfigurability.

B. Distribution of Reconfigurable Logic gates

We apply our approach for $k = 5$ which represents a function of up to 5 variables. While works like [8, 11, 13] have showcased 4 to 5 input logic gates, none of the previous works showcased a 6-input logic gate. Arguments can be made that RFETs support multiple independent gates for a single RFET, but for the present scope, ease of understanding and to be coherent with previous works, we have not considered 6-input standard-cells. While our approach returns 28 such reconfigurable logic gates along with their number of occurrence, not all of them are unique in the context of standard-cell based mapping. Modern technology mappers use NPN-equivalence for mapping to logic gates [1]. For example, logic gate denoted by truth-tables 00010111 (*Minority*) and 11101000 (*Majority*) are equivalent as one can be implemented by the other just by adding an inverter.

The previous experiment gives us a list of reconfigurable standard-cells based on the functional profiling of a benchmark suite. We then use these new standard-cells and report the impact they have on circuit's parameter like area, delay and the number of edges (refers to the total number of connections within a circuit).

1) Realizing Parameters For The Generic Library: While populating the generic library, for area, we consider the number of transistors. Our comparison is only across RFET-based generic libraries. For all the standard-cells, we consider only RFET-based implementations. We use the number of transistors from the calculations done in [20] where all types of RFETs contribute a unit transistor to the area. For a boolean expression in a *Sum-Of-Product* (SOP) form, a minterm consists of literals which are AND-ed together. RFETs can support multiple independent gates on a single channel due to their wired-AND property [18]. Hence, a single RFET can have up to 4 control gate inputs with 1 program gate input. Therefore, a minterm can be mapped to a single RFET. For the present work, we consider only up to 5-input standard-cells as explained before. Hence, parameters of up to 5-input reconfigurable logic gates can be easily calculated. Further, for logic gates, which are binate (for example XOR3) in their literals, we consider inverters for every binate literal while calculating the total number of transistors in a standard-cell. For delay calculation, we use the number of logic-levels between the primary input to primary output. This represents the depth of a logic network.

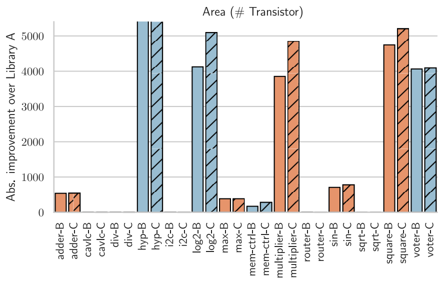
2) Experimental Setup: For technology mapping, we use ABC [4] with three different generic libraries. The baseline generic libray (library-A) is the CMOS-based MCNC generic library proposed with ABC. However, all the logic gates are implemented in RFETs. The second generic library (library-B) is the one mentioned in [17] which is a superset of the library-A containing some hand-crafted reconfigurable logic gates. The third generic library is the superset of library-B with the newly found standard-cells. We use both delay and area-optimized mapping for our experiments.

C. Discussion

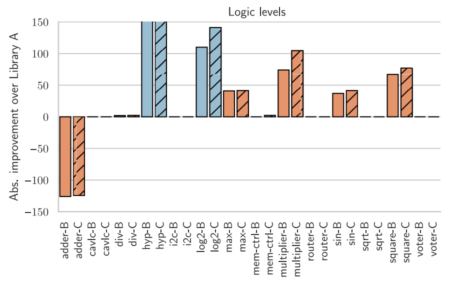
Fig. 4 shows absolute improvement over the baseline for area, delay and edge comparison using our setup with EPFL benchmark suite.

1) Area, Levels and Edges: Fig. 4a and 4b show the area improvement over the baseline for both area and delay-optimized mapping respectively. An important correlation can be made with Fig. 3. The benchmarks which show a higher percentage of reconfigurable cuts also show a higher improvement in area as compared to the baseline. Particularly, with large benchmarks like $multiplier$ and $square$, this high percentage is a good indication that in the presence of such reconfigurable logic gates in the generic library, we got an improved mapping. Further, mapping with library-C shows improvement over library-B. This is due to the new standard-cells which we got during standard-cell extraction. For the hyp benchmark, which is the largest gate, the area improvement comes out to be of the order of $70k$. Hence, in the graph, it is seen going beyond the $y - range$.

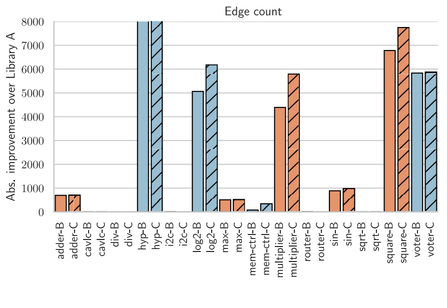
Fig. 4c and 4d show the improvement in the number of logic-levels over the baseline for both area and delay-optimized mapping respectively. The same correlations can



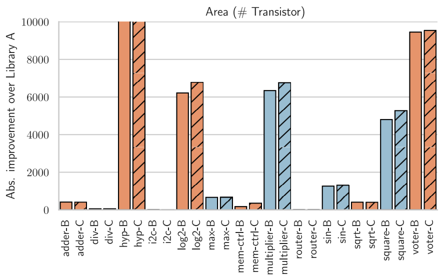
(a)



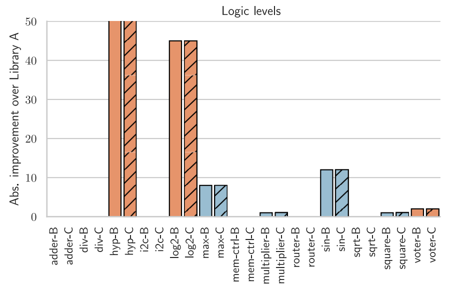
(c)



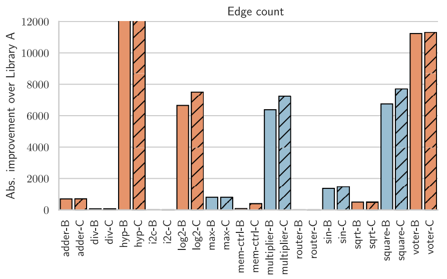
(e)



(b)



(d)



(f)

Fig. 4: (a) Area improvement in area-optimized mapping (b) Area improvement in delay-optimized mapping (c) Improvement in number of levels from primary-inputs to primary-outputs in area-optimized mapping (d) Improvement in number of levels from primary-inputs to primary-outputs in area-optimized mapping (e) Edge improvement in area-optimized mapping (f) Edge improvement in delay-optimized mapping.

be seen with the number of levels as was seen with area. The only exception which can be seen here is for the *adder* benchmark in area-optimized mapping. Since mapping is an intractable problem, ABC chose a more optimal area mapping but compromised on delay as compared to the baseline. Fig. 4e and 4f show the improvement in the number of edges over the baseline for both area and delay-optimized mapping. Edge is an important consideration, because it refers to the number of connections in the circuit. With this result, it can be observed, that with appropriate RFET-based standard-cells, the total number of edges can be reduced. Reduced number of edges directly reflect on the routing cost of the circuit later in the physical synthesis process. Ignoring the results where there are no effect, the average improvement comes out to be 13% in area, 11.5% in delay (or logic-levels) and 16% in the number of edges.

VI. Conclusion

In the present work, we demonstrated an efficient methodology to distill standard-cells which can exploit the reconfigurable nature of emerging nanotechnologies. Through experiments over EPFL benchmark suites, our proposed algorithm generates a list of reconfigurable logic gates. Using these newly extracted reconfigurable logic gates as standard-cells, improvements in area, logic-levels and edge is achieved for both EPFL benchmarks suites.

Acknowledgment

The authors would like to offer special thanks to Steffen Maercker for his valuable assistance. This work is supported in part by the German Research Foundation (DFG) within the Cluster of Excellence "Center for Advancing Electronics Dresden" (CfAED) at the Technische Universität Dresden.

References

[1] Giovanni De Micheli. *Synthesis and optimization of digital circuits*. McGraw-Hill, 1994.
[2] Yu-Ming Lin et al. "High-performance Carbon Nanotube Field-effect Transistor with Tunable Polarities". In: *IEEE Trans. Nano.* (2005).
[3] Satrajit Chatterjee. *On algorithms for technology mapping*. University of California, Berkeley, 2007.
[4] R. K. Brayton and Alan Mishchenko. "ABC: An Academic Industrial-Strength Verification Tool". In: Springer Berlin Heidelberg, 2010.
[5] André Heinzig et al. "Reconfigurable silicon nanowire transistors". In: *Nano Letters* (2012).
[6] M. De Marchi et al. "Polarity control in double-gate, gate-all-around vertically stacked silicon nanowire FETs". In: *IEDM*. 2012.
[7] Tsutomu Sasao. *Switching theory for logic synthesis*. Springer Science & Business Media, 2012.
[8] J. Zhang et al. "Dual-threshold-voltage configurable circuits with three-independent-gate silicon nanowire FETs". In: *ISCAS*. 2013.
[9] J. Zhang et al. "Polarity-Controllable Silicon Nanowire Transistors With Dual Threshold Voltages". In: *IEEE Transactions on Electron Devices* (2014).
[10] L Amarú et al. "The EPFL combinational benchmark suite". In: *IWLS*. CONF. 2015.
[11] J. Trommer et al. "Functionality-Enhanced Logic Gate Design Enabled by Symmetrical Reconfigurable Silicon Nanowire Transistors". In: *IEEE Trans. Nano.* (2015).
[12] L. Amarú et al. "Majority-Inverter Graph: A New Paradigm for Logic Optimization". In: *TCAD* (2016).
[13] J. Trommer et al. "Reconfigurable nanowire transistors with multiple independent gates for efficient and programmable combinational circuits". In: *DATE*. 2016.
[14] M. Raitza et al. "Exploiting transistor-level reconfiguration to optimize combinational circuits". In: *DATE*. 2017.
[15] J. Trommer et al. "Enabling Energy Efficiency and Polarity Control in Germanium Nanowire Transistors by Individually Gated Nanojunctions". In: *ACS Nano* (2017).
[16] S. Rai et al. "Emerging Reconfigurable Nanotechnologies: Can They Support Future Electronics?" In: *ICCAD*. 2018.
[17] S. Rai et al. "Technology mapping flow for emerging reconfigurable silicon nanowire transistors". In: *DATE*. 2018.
[18] M. Simon et al. "A wired-AND transistor: Polarity controllable FET with multiple inputs". In: *DRC*. 2018.
[19] V. Tenace et al. "Logic Synthesis of Pass-Gate Logic Circuits with Emerging Ambipolar Technologies". In: *TCAD* (2018).
[20] S. Rai et al. "Designing Efficient Circuits Based on Runtime-Reconfigurable Field-Effect Transistors". In: *TVSLI* (2019).
[21] *DiSCERN Tool*. URL: https://cfaed.tu-dresden.de/pd-downloads.