Technology Mapping Flow for Emerging Reconfigurable Silicon Nanowire Transistors

Shubham Rai, Michael Raitza, Akash Kumar
Chair For Processor Design, CfAED, Technische Universität Dresden, Dresden, Germany
<firstname.lastname>@tu-dresden.de

***Abstract*—Efficient circuit designs can make use of ambipolar nature of silicon nanowire (SiNW) over CMOS. Conventional circuit Design-Flow fails to use this inherent functional flexibility as CMOS based mapping considers a single logical output from logic gates. To address this, we propose an area-optimized technology mapping which uses this innate reconfigurability, offered by *SiNW* transistors for efficient circuit designs. To enable this objective, we use higher order functions (HOF) to encapsulate this extended functionality. Additionally, the electrical properties of *SiNW* allow us to take advantage of the available inverted forms of fan-ins for additional savings of area for XOR logic family. Experimental results using our technology mapping show that area of *SiNW* based logic design is less by an average of 18.38% as compared to CMOS flow for complete MCNC benchmarks suite. Further, we evaluate our flow for both reconfigurability-aware and static layout for SiNW based logic gates. The whole flow including the new *SiNW* based genlib and the modified ABC tool is made available under open source license to enable further research for any kind of emerging ambipolar transistors.**

I. Introduction

Capable open-source tools for various stages of EDA are imperative for progress in academic and industrial research. With the emerging newer nanotechnology in research, simple and efficient tools become the most important pillar for supporting such endeavors. Silicon Nanowires Reconfigurable FETs (SiNW RFETs) exhibit fully symmetrical p and n-type behavior in a single device as opposed to CMOS technology. The authors in [7] showed that such device characteristics can be exploited to achieve extended functionality. This extended functionality or better *Reconfigurability* has been observed in FPGA or CGRA kind of circuits. However, reconfigurability observed in such systems is due to coarser grain architectural configurations. Silicon nanowire based transistors have brought this reconfigurability at the transistor level. SiNW provides a complementary technology to CMOS by adding more functions per transistors. Further, being a dopant-free technology, RFET integration can be directly implemented onto the back-end of CMOS chip [15].

There are well-developed tools catered for logic and physical synthesis. Logic synthesis in design flow primarily includes minimization, optimization and technology mapping flow for a logic circuit. From 1985, various tools like MIS [5], MIS-MV [8], SIS [11], BDS [17], CUDD [12] enabled logic minimization and optimization. *And Inverter Graph* (AIG) paved the way for ABC [4] which exploited the boolean completeness of AIGs and implemented various algorithms for logic synthesis. Recent works like MIG [2], Amarù et.al. have for the first time tried to target logic synthesis for newer nanotechnologies. Similarly, MixSyn [3] and Xor-Majority-Graphs [6] exploited logic minimization in the circuit by optimizing *XOR* and *AND/OR* logic segments within the same circuit.

None of the above works made use of runtime reconfigurability offered by SiNW RFETS in larger circuits. They primarily used CMOS congruous flow of single logic output from RFET based logic gates in larger circuits. All recent logic synthesis tools mentioned above targeted the logic optimization and minimization of electronic circuits. There was no such attempt to define technology mapping flow for circuits based on silicon nanowire RFET which can later use this exceptional property to choose from a range of functionality from the same device at runtime. Our present work in this particular domain defines efficient technology mapping flow of logic circuits based on silicon nanowire RFETs.

Contributions: The contribution in the present work are:

- We encapsulated reconfigurability with higher order functions, *HOF* for mapping of efficient circuit designs using silicon nanowires RFETs based logic gates. This moves away from the normal CMOS flow in which logic gates have a single immutable function. This enables mapping one of the multiple yet mutually exclusive functions expressed by *HOFs* for reconfigurable aware circuits.
- Based on transfer characteristics of silicon nanowires, we put forth an algorithm to contribute to area savings, by using available inverted forms of fan-ins from the circuit for XOR logic family.
- We propose a novel methodology for technology mapping flow suited for functionally enhanced logic gates based on emerging nanotechnology. The target technology in our case is silicon nanowire but can be extended to any emerging nanotechnology. The new flow potentially can be easily integrated with upcoming logic synthesis flow like [3], [2] to enable reconfigurable aware logic circuits.

In order to elucidate the above contributions, experiments on MCNC benchmarks [18] shows the area numbers for SiNW based logic gates using the above flow after mapping is 18.38% less than as compared to the normal CMOS flow. The whole flow is available online under open source license to enable further research in this domain [13]. The remainder of this paper is organized as follows. Section II gives the background of logic synthesis work and how silicon nanowire as a technology has been used in circuits. Section III describes higher order functions. Section IV mentions the area optimization algorithm through shared inverted fan-ins. This is followed by overview of the flow in section V. Experiments are explained in section VI. Discussions are detailed in section VII. Concluding remarks are given in section VIII

II. Related Work and Motivation

In [15], Trommer et. al. showed efficient and programmable combinational logic gates based on SiNW RFETs. It was shown that because of this extended functionality, a single transistor can replace logic by several transistors. Further, in [10], the authors demonstrated the potential of reconfigurable transistors in a conditional carry adder circuit by showing

978-3-9819263-0-9/DATE18/©2018 EDAA

area and delay gains. These works showed the benefits of SiNW RFETs at the circuit level, MIG[2], MixSyn[3] explored data structures and algorithms to exploit logic minimization and optimization flow. In [1], Amaru et. al. showed that majority functions are natural functional representation for SiNW FETs. In MixSyn, the authors tried to partition and employed different optimization for XOR intrinsic parts of the circuits and *AND/OR* parts of the circuits. This tool explored the XOR implementations offered inherently by ambipolar devices as mentioned in [9]. None of these works truly exploited reconfigurability offered by SiNW and undermine their benefits by using an analogous CMOS flow. While [15] and [10] have demonstrated specific cases of individual gates to simple manual designed circuits, no methodology has been proposed especially for SiNW RFETs.

In this work, we introduce a mapping flow for circuits by utilizing the flexibility in terms of functional output of SiNW RFETs logic gates. Further, we propose a potential saving in the area by reusing inverter logic available in the circuit for XOR-based logic gates. The present work streamlines the technology mapping stage of logic synthesis step so as to enable reconfigurability.

Silicon Nanowire Reconfigurable Transistors

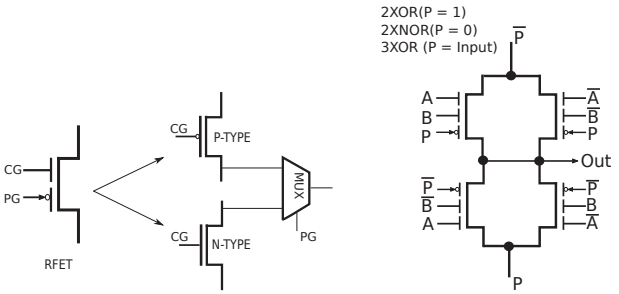
Reconfigurable FETs as a doping-free technology is enabled with sharp metal contacts forming two Schottky junctions at the source and drain as shown in Fig. 2. One can notice the V-shaped I-V characteristics demonstrating both p and n functionality from a single device. The gate overlapping with the drain is called the Program Gate (PG) and determines the overall behavior of Silicon nanowires transistors. The Control Gate (CG) overlaps with the source and controls the carrier flow [15]. An important property of these devices, that unlike CMOS the on-current resistance does not depend on the channel length but it is dominated by source-sided Schottky Barrier [16]. These facilitates adding extra gate terminals over the nanowire channel without loosing performance. Hence, multi-gate extensions of RFETs like three input gate FETs (TIGFETs) and multi-input gate FETs (MIGFETs) are possible [19]. Interesting extensions of such polarity-controlled devices reported in recent works [14] include germanium based nanowires which show better threshold voltages and higher normalized on-currents as compared to existing nanowire technology. These emerging devices with greater potential on offer, require a feature-rich design flow (both logic and physical synthesis) which can make use of enhanced properties of such newer devices.

III. Encapsulating Reconfigurability through Higher Order Functions (HOF)

Reconfigurability is achieved in SiNW at runtime by changing the polarity of the program gate terminals. In terms of mathematics, reconfigurability can be encapsulated using a *Higher Order function* as described in the following equation:

$$f(x,y,z,w) = g(x,y,z), when\ w = 0$$
$$= h(x,y,z), when\ w = 1$$

Here f is a *HOF* of four variables. Functions g and f are two different functions. In the above expression, f can be represented in terms of functions g and h depending upon the values of w. Function f can be seen encapsulating functions g and h. Analogous to the above mathematical function, SiNW



(a) *SiNW* RFET **(b)** XOR based on RFETs

Fig. 1: SiNW RFET based logic Gate and comparison with CMOS

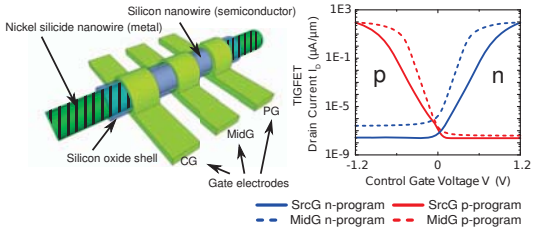
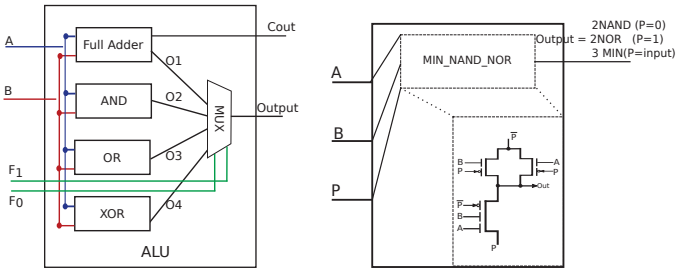


Fig. 2: Silicon Nanowire three independent gate (TIGFET). The graph shows symmetry in p and n-type behavior for *SiNW* RFET [10]

RFET has been shown to behave as p-type and n-type and that choice can be made by changing the potential of Program gate. The program gate is the w of the above function. This can be very well represented by the Fig. 1a. A SiNW RFET with two gate terminals can be easily be represented as a *NMOS* and a *PMOS* whose outputs are going to a *2to1 MUX*. The select line of that mux is basically the PG terminal of a RFET. Functionally, we can say that a RFET is a higher-order representation of the two kinds of MOSFETs, followed by a MUX. Adding to that, since electrical property can change by changing PG, this brings runtime reconfigurability similar to a MUX.

Such mathematical concept is not new in electronics, and an **Arithmetic Logic Unit (ALU)** is a perfect example of such systems where a user has an option to choose from a set of arithmetic functions. The whole ALU function, can be seen as a function f encapsulating functions like *addition*, *multiplication* as g and h respectively. This kind of reconfiguration is available because of extra circuitry and nice part of such circuit is that you have multiple control paths which can give more than one function simultaneously. We term this reconfigurability as **extrinsic reconfigurability**. This is shown in Fig. 3a. We can see that each inbuilt logic function does produce multiple outputs through multiple control paths shown as O1, O2, O3 and O4. At the end, the *MUX* is used to select the required output. We can see from the figure that logic functions from each logic gates are implemented independently and their selection is done by MUX at runtime.

The enhanced functionality exhibited by SiNW RFET differs from that of a configurable circuit like ALU as there is a mutual exclusion in the availability of multiple functions for SiNW RFETs. By mutual exclusion, we mean that a RFET based logic gate can have one and only one logical output at a single instant of time unlike extrinsic reconfigurability. This mutual exclusion is the result of the the way input variables are



(a) Extrinsic Reconfigurability **(b)** Intrinsic Reconfigurability

Fig. 3: Higher Order Functions explained

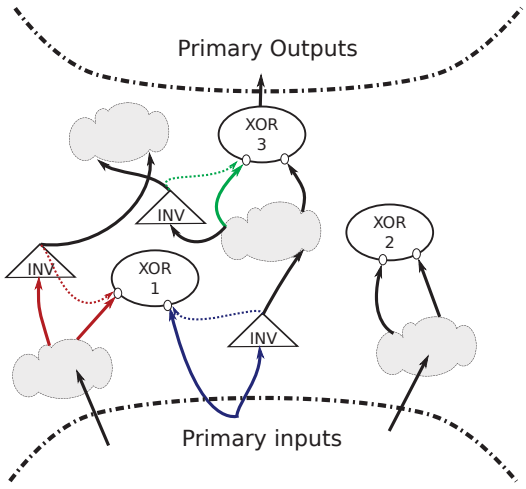


Fig. 4: *Area optimization using Inverter Adjustment*

connected to the gate terminals of the *HOF*. This is shown in Fig. 3b. Unlike extrinsic reconfigurability, there is neither extra circuitry involved here, nor are there multiple output paths followed by a selection procedure. The unique behavior is possible due to electrical properties of SiNW RFETs. We term such kind of reconfiguration as **intrinsic reconfigurabilty**. An example of such logic gate is given in Fig. 1b. It represents a four transistor structure. It can behave as a *XOR* when $P = 1$, a *XNOR* when $P = 0$ and a *3-XOR* function when *P* is the third input.

Extending this transistor behavior, we have logic gates which can show multiple functionalities from the same structure as mentioned in [15]. Such instances of logic gates which show multiple functionalities can be termed as **higher order functional (HOF)** Gates. While it is possible to realize *HOFs* in CMOS, they are generally made of higher number of transistors. This concept is novel in SiNW RFETs, as opposed to CMOS because in CMOS, where there is only one gate terminal, and any configuration of inputs will ultimately give a single logic output. Such expressive power of intrinsic reconfigurability can be very well represented using *HOF* gates. We use *HOF*s for mapping of reconfigurability aware electronic circuit.

IV. Area Optimization through Sharing of Inverted Fan-ins

An important difference to note among the logic gates proposed in [15] are the XOR family of logic gates. In both renditions of the XOR gate i.e. *2-bit XOR* or *3-bit XOR*, the complemented and actual forms of each input are required in the logic gate. So in order to calculate the actual number of transistors involved in these logic gates, we must include an inverter for getting the inverted phase of each input within the logic gate boundary. Like in XOR2_XNOR we have *A* and *B* required in both the phases. Hence, overall area for the XOR-based logic gates would increase by a factor of $Nos\ of\ input\_variables$ x $2(RFETs)$. This is an overhead which we have to take into account.

However, if we are using the XOR logic at multiple places in a logic circuit and if some of the fan-ins are available in the inverted form within the logic circuit, then that can lead to the redundant use of inverters. Hence, we explore the feasibility of harnessing such inverted forms of fan-ins available in the circuit. An obvious problem arises as complemented forms of fan-ins available from other parts of the logic circuit add to the fanout delay due to the longer length of metal wires. We solve this problem by utilizing the unique electrical properties of silicon nanowire multi-gate RFETs. XOR family of logic gates uses interior gate terminals of multi-input RFETs. Authors in [10], states that the gate terminals placed above the Schottky barrier in RFETs are faster gate terminals. In Fig. 2 I-V characteristics of interior gate terminals are represented by dotted lines exhibit steeper slope for transitions. In XOR both these inputs are fed to these faster gates terminals. Therefore, the faster interior gate terminals compensate the delay caused by the long length of metal wires. Hence, in cases of logic circuits which have these inputs already available in inverted forms in other parts of the circuit, it is feasible to use such inverted forms for these logic gates thereby reducing the overall number of transistors required in the circuit.

Fig. 4 illustrates area optimization in logic network. The cloud-shaped part represents the combinational parts of a logic network. There are three XOR nodes whose fan-ins are shown along with a bubble, signifying inverters required for complemented input. Consider the XOR node 1 which has one fan-in coming from the primary input (blue arrow) and other coming from the left combinational part(red arrow). However, we can see that inverted forms of both the fan-ins are available in the circuit and hence a suitable scenario for area optimization is possible here as shown through dotted lines. In this case, our algorithm will get away with both the internal inverters in the XOR gate. Similarly, for XOR node 3, one of the fan-ins are available in inverted form (green arrow) and that too contributes to area saving. XOR node 2 has none of the fan-ins available from the circuit and will have inverter contribution for both the fan-ins. Logic networks using XOR-based logic family can tap this kind of optimization to reduce the overall area. Further, limiting the inverter sharing for a certain number of fan-ins is required to prevent higher fanout delays, if a single inverter output is shared by multiple XOR gates. Such logic sharing during the technology mapping also eases out later stages of physical design. Circuit designers can use such optimization scheme and can predict the overall use of inverters in the final layout of the circuit.

To calculate the number of such available inverted signals, we refer to the pseudo code shown in Algorithm 1. Initial area during mapping is calculated in $Area$. If a particular node matches with the XOR family, we look for available inverted forms of fan-ins in the overall circuit. An area adjustment is required at the same time for every available fan-in. The

Algorithm 1 Area Savings using already available inverted fanins

Input: Mapped Network G
Initial Area of a Mapped network = $Area$
while Iterate over each node in G **do**
  if Node.Match == XOR Family **then**
    while iterate over each node in G **do**
      if Node Fanin available in inverted forms **then**
        Mark such fanins
      end if
    end while
  end if
  Adjust the final Area $Final\_Area$
end while
if $Area < Final\_Area$ **then**
  Check for other mappings of G
end if
Output: Choose the mapping with the min area

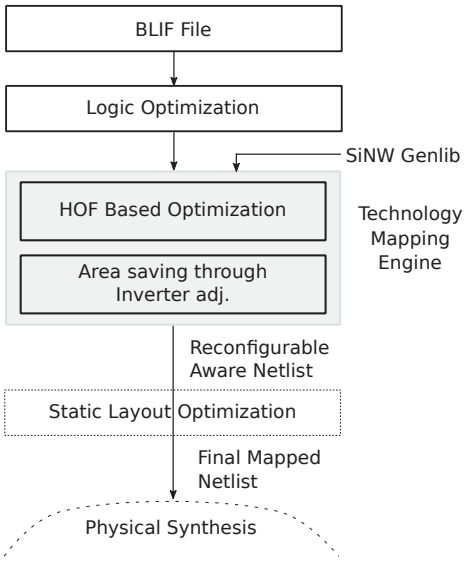


Fig. 5: Entire technology Flow suited for SiNW RFETs

mapper finally gives that mapping, which has the lowest area of the overall circuit. Instances of XOR logic gates which does not find such available fan-ins, contribute with their default area.

V. Technology Mapping Flow for RFETs

Fig. 5 shows the complete flow of our technology mapping for logic design based on SiNW Reconfigurable FETs. The output after the logic optimization is fed to our technology mapper. The higher order functions encapsulate reconfigurable logic gates based on silicon nanowires RFETs. At first, the mapping is carried out using higher order functions followed by area optimization using inverter adjustments. The output of this technology mapping is a netlist which is basically a reconfigurable-aware logic circuit. Until this point, each component in the circuit is a HOF and can exhibit runtime reconfigurability depending upon input P at program gate for each logic gate [15].

TABLE I: Genlib for *RFETs*

Name of the gate	identifiers.	Nos. of fn.	Area	1st fn. (HOF)	2nd fn., 3rd fn
inv1			1	O=!a	
nand2_ nor2_ min3	sinw	3	2.625	O1=((!(a*b)) * !c) + ((!(a+b)) * c)	O2 =!(a*b), O3 = !(a_b)
xor2_ xnor2_ xor3f	sinw inv_adj	3	5.5	O1=((a^b) * !p) + (!(a^b) * p)	O2=a ^b O3= !(a^b)
mux			4.5	O=(!s*a)+(b*s)	
aoi_oai21	sinw	2	4	O1=!(a*b+c)	O2=!((a+b)*c)
emux			8.5	O=((!(a*c + (!c)*b))*e) + (!(a*d + (!d)*b)*(!e))	
nand3_nor3	sinw	2	3.25	O1= !(a*b*c)	O2 = !(a+b+c)
maj_min	sinw	2	4.75	O1= ((a*b)+(b*c) +(c*a))	O2=!((a*b)+(b*c)+(c*a))
xor3_xnor3	sinw inv_adj	2	9.75	O1=(a^b^c)	O2 = !(a^b^c)

After this, the user has an option to choose a reconfigurability-aware layout or a static layout. Static layout indicates that the logic gate's functionality is defined by the layout and runtime reconfigurability cannot be used. For runtime-reconfigurability, the reconfigurable-aware netlist can be left as it is with all the P's of each logic gate as inputs, to be defined by the circuit designer at runtime. For the static layout, the P of all logic gates have to be fixed to either V_{dd} or V_{ss} so that a logic gate loses its runtime-reconfigurability and behave as CMOS analogous logic gate. In this case, the logic circuit gains in terms of area with fewer transistors. Circuit designers have to take care of this trade-off between the number of transistors and reconfigurability with their designs. After this decision, the netlists can undergo through physical synthesis process for the final circuit.

VI. Experiments

In this section, we present our experimental setup for comparing area post reconfigurability-aware technology mapping for both SiNW and CMOS technology. The area in terms of numbers of transistors is shown for both the flow.

A. Silicon Nanowire RFETs based Genlib

To enable ABC tool to support multiple functionalities for SiNW RFETs logic gates, genlib has to be modified to support the above mentioned changes. Since SiNW has multiple functionalities, a logic gate entry in genlib has to updated with the number of outputs it can support and a description of those outputs. In order not to interfere with the normal flow of ABC, we have used **sinw** identifier in genlib. The area numbers are carefully calculated taking consideration of all the factors. Since the area here is basically the number of transistors, hence a RFET with one control gate and one program gate is considered as 1 transistor. Hence, an inverter has 2 RFETs and just like the CMOS flow, the area of the inverter is considered as 1. All other logic gates are normalized with respect to the area of the inverter. Since other logic gates have TIGFETs and MIGFETs, we have to consider the area occupied by these multi-gate FETs. This kind of consideration has been missing from previous works in this domain. The contributions of multi-gate FETs are taken according to Table II. To enable this area optimization due to inverter sharing in ABC tool, we have added a the **inv_adj** identifier in the genlib to the gate definition.

TABLE II: Area for multi input Gates FETs

FET type	PG + CG	Area
RFET	1 + 1	1
TIGFET	1 + 2	1.25
MIGFET	1 + 3	1.50

TABLE III: Time and Area Numbers between CMOS and SiNW RFETs

Bench-marks	Time C (in sec)	Time S(in sec)	Time SS (in sec)	Area _C	Area _S	Area _SS	inv Adj. SS	Inv. Adj. S
C432	0.01	0.01	0.01	416	482	351	16	17
C1355	0.02	0.03	0.03	752	714	626	112	15
C1908	0.01	0.02	0.01	725	804	644	67	12
C2670	0.02	0.02	0.02	1309	1382	1034	38	28
C3540	0.02	0.03	0.02	1889	2398	1828	32	31
C6288	0.13	0.18	0.16	4429	3063	2755	218	203
C7552	0.06	0.1	0.08	3842	3870	3162	170	155
t481	0.05	0.04	0.05	2445	3025	2119	0	0
rot	0.03	0.03	0.03	1110	1357	981	13	13
b9	0.02	0.03	0.02	184	230	173	0	0
dalu	0.05	0.06	0.05	2382	2941	2335	17	39
des	0.11	0.11	0.11	7619	8344	6171	200	111
k2	0.06	0.05	0.05	3076	3735	2544	0	0
Avg.	**0.05**	**0.05**	**0.05**				**67.92**	**48.00**

In Table I, for *nand_nor_min* the number 3 is the number of functions this logic gate can offer which are defined by $O1$, $O2$ and $O3$. Then 2.625 is the area dependent upon the number of RFETs, TIGFETs and MIGFETs normalized to the area of an inverter. We are not taking the delay in the calculations of present work. There are 8 different logic gates in this genlib which are defined by the functions each can support. In [10], the authors proposed an E-MUX to replace a two-stage MUX with a single logic gate which further enabled reduced area and delay numbers. We have included the *E-MUX* logic gate in our genlib. The present work provides a simple mechanism to include and analyze the benefits of such futuristic combinational logic gates.

To study static layout optimization as shown in Fig. V, we carried out experiments using a modified genlib. We assume beforehand that a particular logic gate has a single functional output i.e. no reconfigurability. By assuming this, we can get away with the inverter area within the logic gate boundary required by program gate terminal of a RFET. Unfortunately, with this change, we will loose *HOFs*. From the layout point of view, we have to maintain the Vdd and Vss at P and $\overline{P}$. Actual layout can be decided during physical synthesis. Thus, in almost (except *MUX* and *EMUX*) all the logic gates the area is made one unit less. For example in nand_nor_min3 logic gate, the area assumed for the first case is $(1.25+2+2)/2 = 2.625$. The new area would be 1.625. Similar adjustments are made in all the logic gates and the area is calculated again for the same set of benchmarks. It is to be noted that this reduction will not be incremental for XOR family logic gates, as the inputs are still required in both normal and complemented forms apart from the program gate. This would be analogous to CMOS flow with all the logic gates having unitary functional output.

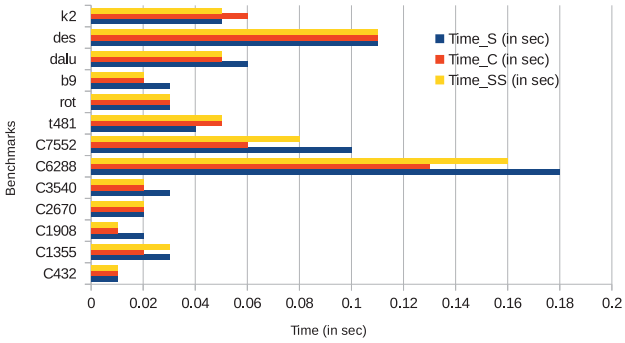


Fig. 6: *Time Taken by Various Flow*

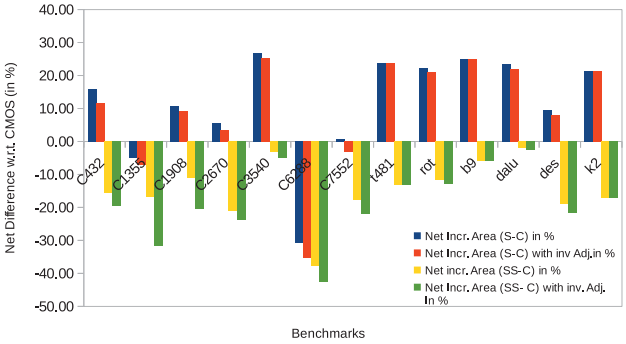


Fig. 7: *Area Numbers Comparison for Various Flow*

B. Experimental Setup

The optimization for mapping with hof based logic gates is done in ABC tool. We have used the ABC tool as it's the state-of-the-art logic synthesis tool with a well-developed covering and mapping flow. All the standard optimizations in ABC are intact and there is no interference with the default flow. In our experiments we have just used the ABC commands *strash* and *balance* of the logic network. For timing calculations, we used *time* command after *map* command. The area numbers are through *print_gates* commands which also list the adjustment due to the availability of inverted fan-ins. The benchmarks used in our experiment are the MCNC benchmarks. The area numbers for both the flows are compared post the mapping phase.

VII. Results and Discussion

Results for the three flows on MCNC benchmarks are listed in Table III. Due to restriction in space, we are showing the results for benchmarks *suggested for combinational multi-level set* [18]. The letters C, S and SS refer to the CMOS flow, SiNW reconfigurability-aware flow and SiNW static flow respectively.

A. Time Comparison

The columns in Table III– *Time_S*, *Time_C*, *Time_C* refers to the time taken by the three cases in seconds during mapping by ABC tool. From the graph in Fig. 6 it is clear that in almost all the cases, the time taken by the new mapper is similar to the CMOS genlib based default mapping. In some cases, as in **C6288**, the time taken for Silicon Nanowire flow

is more because the mapper uses a large number of XOR family gate (260 in this case). The XOR gates enables the area saving due to inverter adjustments which takes some iterations. Hence if the circuit contains a large number of XOR nodes, the time taken by the mapper increases slightly. In contrast, the benchmark t481 takes less time than CMOS flow because the mapper doesnot use any XOR family gates. The final averages of 219 testcases are 0.030, 0.033, 0.031 seconds respectively for CMOS flow, SiNW reconfigurability-aware and SiNW static flow. Hence, the new mapping is similar to the contemporary CMOS mapping.

B. Area Comparison

In Table III, the columns *Area_C*, *AREA_S* and *Area_SS* are the areas number for three runs computed by *print_gates* command in ABC. We calculated the net difference with respect to the CMOS flow for the SiNW Flows as shown in the Fig. 7 *Net increase in Area (S-C)%* and similarly *Net increase in Area(SS-C) %*. Another parameter is to include the area saving due to the availability of inverted fan-ins in case of XOR family gates which are represented in respective columns as *Inv. Adj.* and *Inv. Adj. SS*. The net difference with respect to CMOS flow including inv. adjustment is also calculated. From the positive averages we can see that due to reconfigurability, there is an obvious area overhead. That can be also guessed since all the logic gates mentioned in Table I has the larger area than their corresponding logic gates in MCNC.genlib. That is true as well considering the fact that reconfigurability will surely have an area overhead.

From the graph in Fig. 7, we can notice some benchmarks are anomalies as they have lesser area as compared to CMOS. That is true when the mapper uses more higher order functions to match nodes of the logic circuits. Further, if the circuit has more XOR based family and if inverted fan-ins are available, then there would be greater area savings. Some of the mappings in the complete 219 benchmarks, also used emux (the new gate we added in genlib) and that leads to extra saving as it is 5 fanin gate.

The final average net difference of area w.r.t to CMOS for complete 219 testcases are 17.48% and 16.25% for mapping of reconfigurable SiNW logic Gates. The numbers come down to -13.76% and -15.05% for mapping of static logic Gates. For the benchmarks in Table III, net difference comes out be 11.95% and 9.62% for reconfigurability-aware and -14.81% and -18.34% . The average area saving due to sharing of inverted fan-ins for complete benchmark suite is 8.2 for reconfigurability-aware flow and 9.26 for static flow. Further, this static flow will greatly reduce the routing area as the logic gates can be changed such that the program gate and the source and drain are connected to the *VDD* and *VSS* within the gate boundary. The above-mentioned flow can act as a predecessor to pass information for physical synthesis flow.

An important feature of the above algorithms for mapping is that it can be potentially extended to any other logic synthesis tool. This means that for tools like MIG [2], Mixsyn [3] which were basically extending abstraction for emerging technologies and have shown better results as compared to ABC in logic synthesis can be coupled with our algorithm.

VIII. Conclusion

In the present work, we demonstrated a modified technology mapping flow which enables the use of reconfigurable logic gates based on SiNW. We included multi-input gates in our technology mapping flow to show their benefits in larger range of circuits. Through experiments, a comparison between static and reconfigurable-ready technology mapping was done in terms of area numbers keeping the logic optimization and minimization steps intact. For flows in which reconfigurability is not enabled, we get better area numbers (on an average less 18.38%) with SiNW technology as compared to CMOS. Performance of the new mapping flow was also evaluated and was found to be similar to the default mapping flow of ABC. Useful concepts to encapsulate reconfigurability using *Higher Order Functions* were also elaborated. A mechanism to contribute to area saving was presented for XOR-based logic family using already available inverted forms of fan-ins. In this paper, we have shown the technology mapping of reconfigurable SiNW logic gates for larger circuits. The flow described here is not limited to SiNW RFET but can be extended to any emerging reconfigurable nanotechnology.

Acknowledgment

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] L. Amarù, P. E. Gaillardon, and G. De Micheli. "Efficient arithmetic logic gates using double-gate silicon nanowire FETs". In: *NEWCAS*. 2013.

[2] L. Amarú, P. E. Gaillardon, and G. De Micheli. "Majority-Inverter Graph: A New Paradigm for Logic Optimization". In: *TCAD* (2016).

[3] L. Amarù, P. E. Gaillardon, and G. De Micheli. "MIXSyn: An efficient logic synthesis methodology for mixed XOR-AND/OR dominated circuits". In: *ASP-DAC*. 2013.

[4] R. K. Brayton and Alan Mishchenko. "ABC: An Academic Industrial-Strength Verification Tool". In: ed. by Tayssir Touili, Byron Cook, and Paul Jackson. Springer Berlin Heidelberg, 2010.

[5] R. K. Brayton et al. "MIS: A Multiple-Level Logic Optimization System". In: *TCAD* (1987).

[6] W. Haaswijk et al. "A novel basis for logic rewriting". In: *ASP-DAC*. 2017.

[7] André Heinzig et al. "Reconfigurable silicon nanowire transistors". In: *Nano Letters* (2012).

[8] L. Lavagno et al. "MIS-MV: optimization of multi-level logic with multiple-values inputs". In: *ICCAD*. 1990.

[9] M. De Marchi et al. "Polarity control in double-gate, gate-all-around vertically stacked silicon nanowire FETs". In: *2012 International Electron Devices Meeting*. 2012.

[10] M. Raitza et al. "Exploiting transistor-level reconfiguration to optimize combinational circuits". In: *DATE*. 2017.

[11] E. M. Sentovich et al. "Sequential circuit design using synthesis and optimization". In: *ICCAD*. 1992.

[12] Fabio Somenzi. "CUDD: CU decision diagram package release 2.3. 0". In: *University of Colorado at Boulder* (1998).

[13] *Technology Mapping Flow for Reconfigurable Silicon Nanowire FETs*. URL: https://cfaed.tu-dresden.de/pd-downloads.

[14] J. Trommer et al. "Enabling Energy Efficiency and Polarity Control in Germanium Nanowire Transistors by Individually Gated Nanojunctions". In: *ACS Nano* (2017).

[15] J. Trommer et al. "Reconfigurable nanowire transistors with multiple independent gates for efficient and programmable combinational circuits". In: *DATE*. 2016.

[16] W. M. Weber et al. "Non-Linear Gate Length Dependence of On-Current in Si-Nanowire FETs". In: *European Solid-State Device Research Conference*. 2006.

[17] Congguang Yang and M. Ciesielski. "BDS: a BDD-based logic optimization system". In: *TCAD* (2002).

[18] S. Yang. *Logic Synthesis and Optimization Benchmarks User Guide: Version 3.0*. 1991.

[19] Jian Zhang, Pierre Emmanuel Gaillardon, and Giovanni De Micheli. "Dual-threshold-voltage configurable circuits with three-independent-gate silicon nanowire FETs". In: *ISCAS*. 2013.