Architecture Exploration and Delay Minimization Synthesis for SET-Based Programmable Gate Arrays

Chia-Cheng Wu
Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan
Email: s104062522@m104.nthu.edu.tw

Kung-Han Ho
Institute of Electronics
National Chiao Tung University
Hsinchu, Taiwan
Email: kunghanho@gmail.com

Juinn-Dar Huang
Department of Electronics Engineering
National Chiao Tung University
Hsinchu, Taiwan
Email: jdhuang@mail.nctu.edu.tw

Chun-Yao Wang
Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan
Email: wcyao@cs.nthu.edu.tw

***Abstract*—Power consumption has become a primary obstacle for circuit designs at present. Single-Electron Transistor (SET) at room temperature has been demonstrated as a promising device for extending Moore's law due to its low power consumption. Since, only a few electrons are involved in the switching process, the drivability of SETs is ultra-low such that the height of an SET array is limited to a small number. This paper presents a delay minimization synthesis flow that decomposes a circuit into a network of SET Array Blocks (SABs) with a fixed height and width. The experiments were conducted for different sizes of SABs over a set of benchmarks. The experimental results showed that we can have the smallest average Area Delay Product (ADP) when the height is 5 and the width is 10 of an SAB, which indicates that such size of SABs is proper to synthesize SET networks.**

***Keywords*—Single-Electron Transistor (SET); SET Array Blocks (SAB); delay minimization synthesis flow;**

I. Introduction

As CMOS technology nodes are continuously scaling down, power consumption has become a primary concern in electronic system design [27]. To deal with this issue, various emerging low-power devices have been proposed in recent years. Among these low-power devices, Single-Electron Transistor (SET) is considered as one of the promising candidates. Some demonstrations of SET devices have proved that SET devices can operate at room temperature with only a few electrons [22][23][26].

Since only a few electrons are involved in the switching process, SETs suffer from low transconductance. Therefore, the traditional CMOS architecture is not applicable to SETs. To this end, a Binary Decision Diagram (BDD)-based architecture [2] was proposed as a practical approach to implement logic functions on SETs [1]. Furthermore, the BDD of any Boolean function can be used to map onto a BDD-based SET array [11][15][16]. A BDD-based SET array is a hexagonal nanowire network, which is composed of a set of node devices. Each node device has one entry branch and two exit branches as shown in Fig. 1(a). Fig. 1(b) illustrates that electrons enter a node device via the entry branch and pass through either the left or the right exit branch depending on the control signal of the wrap-gates. An exit branch is a segment of SET controlled nanowire and has four operational modes: *open*, *short*, *active-high*, and *active-low*. In this hexagonal network, each row of the network is controlled by the same input variable. Therefore, a given function represented by BDD can be simply mapped onto such hexagonal network. The first BDD-based hexagonal nanowire network has been demonstrated in [16].

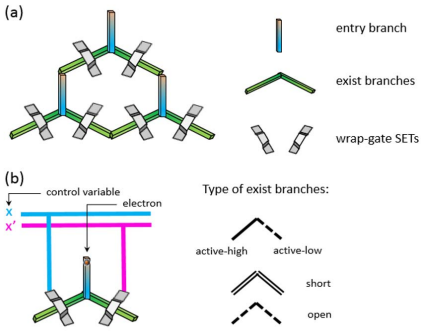

Fig. 1. (a) Physical structure. (b) Logic representation of a node device

However, the proposed BDD-based architecture in [16] is not amendable to functional reconfiguration. Furthermore, the entire circuit becomes useless if there exists a defective nanowire segment or any defective SET on it. Fortunately, a reconfigurable SET array was proposed to deal with these problems [10]. The automatic synthesis methods of reconfigurable SET array targeting at area minimization were proposed in [5][6][25]. Since the height of an SET

This work is supported in part by the Ministry of Science and Technology of Taiwan under Grant MOST 106-2221-E-007-111-MY3, MOST 103-2221-E-007-125-MY3, and MEDIATEK Research Center Doctoral Talent Fellowship.

2159-3477/18/$31.00 ©2018 IEEE
DOI 10.1109/ISVLSI.2018.00055

array is equal to the number of input variables of the given Boolean function, minimizing the area of an SET array means minimizing its width [8][18][19]. A recent work proposed an area minimization synthesis flow for reconfigurable SET arrays, which considered two fabrication constraints in the early stage [3]. The method minimized the number of product terms, which were extracted from a given Reduced Ordered Binary Decision Diagram (ROBDD), by dynamically shifting variables, and modeled the product terms ordering as the Traveling Salesman Problem (TSP) [4].

The aforementioned SET synthesis algorithms can map a function onto an SET array. However, there are more than one hundred inputs in some functions. Since the operations in SET arrays involve only a few electrons, the height, that is corresponding to the number of inputs, of an SET array is limited. The work [20] perceived this issue and conducted device-level experiments. The experimental results indicated that the height of an SET array is about 10. This design consideration issue has been addressed in [12] such that the height of a synthesized SET Array Block (SAB) was limited to 10. However, the experimental results in [12] showed that SABs have various widths with respect to different functions. On the other hand, the verification method on the synthesized SET array was proposed in [7]. The defect issue, diagnosis and mapping for defective SET arrays were also addressed in [13][14][17] recently.

In this paper, we use the concept of Field Programmable Gate Arrays (FPGAs) to divide a complex circuit into a multi-level network consisting of SRAM-based blocks. An SRAM can be considered as a Lookup Table (LUT). A 32-bit SRAM, for example, can be considered as a 5-LUT, which can implement any function within five variables. The limited number of inputs of each LUT, K, is similar to the limited height of an SET array. However, the width of an SAB is also considered as a constant number W. The SABs we used in this work with K inputs and W width are called (K, W)-SABs. There are two objectives of this work. The first one is to develop a delay minimization synthesis algorithm for decomposing a complex circuit into (K, W)-SABs. The other objective is to explore the proper parameters, K and W, for SET-based programmable gate arrays. We conducted experiments over a set of benchmarks with different sizes of SABs. The details of the experiments will be shown in Section IV.

The rest of this paper is organized as follows. In Section II, we introduce the architecture of reconfigurable SET arrays, and the background of this work. Section III presents our synthesis flow. Section IV shows the experimental results. The concluding remarks are given in Section V.

II. Preliminaries

A. Reconfigurable SET Array

The structure of a reconfigurable SET array is shown in Fig. 2(a). A reconfigurable SET array consists of three layers, which are SET device layer, configuration layer, and input signal layer from the bottom to the top, respectively. The SET device layer is a hexagonal network that is constructed by identical SET nodes. The configuration layer determines the operational mode of each node in an SET array. By providing different voltage biases, an SET node can be set in three operational modes: 1) active; 2) open; and 3) short, as shown in Fig. 2(c). The input signal layer is an interface between the input signals and the SET nodes. An input signal can determine whether the corresponding active SET nodes are ON or OFF. Due to the limitation of the SET array structure, the SET nodes in the same row are controlled by the same input signal.

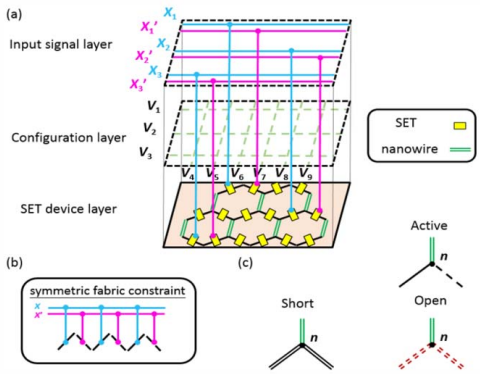

Fig. 2. (a) Architecture of reconfigurable SET arrays. (b) Symmetric fabric constraint. (c) 3 types of operational modes.

B. Symmetric Fabric Constraint

Symmetric fabric constraint reduces the wiring area of SET arrays by having an input signal x and its complementary x' connected to left edges and right edges, or vice versa, of SET nodes in the same row [10], as illustrated in Fig. 2(b). This constraint enforces that a pair of left and right edges of a node need to be one of (*high*, *low*), (*low*, *high*), (*short*, *short*) and (*open*, *open*). If an SET node is in the active operational mode, the pair of edges of the node can be either (*high*, *low*) or (*low*, *high*), which means that an input variable x is connected to the left (right) edge and the complementary x' is connected to the right (left) edge. If an SET node is in the short (open) operational mode, both edges are short (open).

C. Product-Term-Based Mapping Approach

The product-term-based approach synthesizes SET arrays for each Primary Output (PO) of a given Boolean function. The approach configures one path on the SET array for each product term of a PO without creating any invalid paths. The mapping rule is to configure *high* for bit 1, *low* for bit 0, and *short* for don't care. Fig. 3 shows an example of the mapping procedure. We follow the mapping rule to configure or to reuse an edge for each bit of p_0 from the top to the bottom of an SET array. The mapping result is demonstrated in Fig. 3(a). For mapping p_1, the first two configured rows can be reused since the first two bits of p_0 and p_1 are the same. Then the path branches at the coordination of $(x, y) = (0, 2)$, as illustrated in Fig. 3(b).

D. Mapping Flow for SET Network

Since the work [20] brought the issue of height constraint in the SET array mapping, decomposing a large SET array into a network of SABs becomes more important. Fig. 4 illustrates the flow of SET network mapping proposed in [12]. It translates a given Boolean circuit that is represented by an And-Inverter Graph (AIG) into a BDD network, then maps the network onto an SET network. The mapped SET network consists of SET arrays with the fixed height but different widths. The width of each SET array is various with respect to the corresponding sub-function it mapped.

E. Problem Formulation

Given the AIG representation of a Boolean network and user-defined parameters K and W, we map the netlist into a set of (K, W)-SABs such that the depth (delay) of the SABs is minimized.

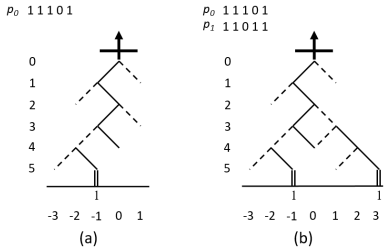

Fig. 3. Example of product-term-based mapping procedure. (a) p_0. (b) p_0+p_1.

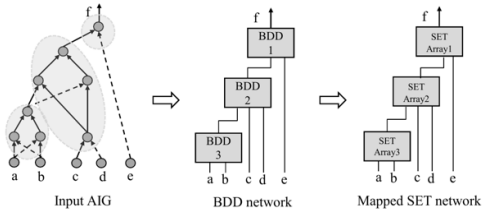

Fig. 4. The flow of mapping an SET network in [12].

III. Proposed Synthesis Algorithm

In this section, we present our delay minimization algorithm for decomposing a large reconfigurable SET array. To simplify the measurement of delay in the mapped SET network, we assume that the delay of each SAB is identical. Therefore, measuring delay of a mapped SET network is to find the longest path in the network consisting of SABs. As mentioned in Section I, the main concept of the proposed synthesis algorithm is similar to the one targeting LUT-based FPGA. An LUT-based FPGA flow decomposes a Boolean network by the cut enumeration [9], and then packs sub-functions into LUTs. Our synthesis flow is divided into two phases: Phase 1 is to generate a set of cuts for each node from the Primary Input (PIs) to the POs in the AIG under the height constraint, and to label the depth and area of every cut and node. The labeling helps determine the mapping priority of cuts in the cut set. Phase 2 is to map sub-circuits from the POs to the PIs to generate the SAB network.

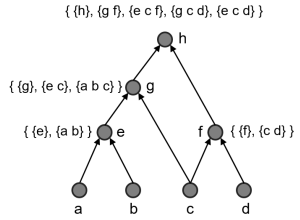

Fig. 5. An example of cut enumeration with K = 3.

A. Phase 1: Cut Generation and Node Labeling

Our algorithm decomposes the given AIG into K-bounded sub-networks, where the number of fanins in a sub-network does not exceed K. The cut enumeration [9] generates all K-feasible cuts for each node in the AIG from the PIs to the POs. A cut C of a node n is a set of nodes in its transitive fanins such that any path from a PI node to node n must pass through at least one node in the cut. A cut is said to be K-feasible if the size of the cut, which refers to the number of nodes in the cut, is no more than K. Fig. 5 illustrates an example of cut enumeration with K = 3. The three cuts, $\{g\}$, $\{e\ c\}$, and $\{a\ b\ c\}$ belong to the node g. The K-feasible cut of any PI node contains only the PI node itself. For a non-PI node v, we apply the cut generation function $\oplus^K$ on the cut sets a and b of its fanins to generate a cut as follows:

$$a \oplus^K b = \{\ x\ \cup\ y \mid x \in a,\ y \in b, \mid x\ \cup\ y\ \mid \leq K\}$$

Since this cut generation follows the topological order of the network from the PIs to the POs, it guarantees that the cut sets a and b have been generated before generating the cut set of node v.

During the cut enumeration, every node will have a label that contains the node depth $D(n)$ and the node area $A(n)$. For each cut, we also label the cut depth $D(c)$ and the cut area $A(c)$ to determine the mapping priority of cuts in the cut set. The cut depth, $D(c)$, is the largest node depth of this cut, which is equivalent to the level from the PIs to the cut, as shown in Eq. (1). The cut area, $A(c)$, is referred to the summation of node area of nodes in this cut, and it represents the number of SABs we need when constructing the cut, as shown in Eq. (2). The node depth of a node n, $D(n)$, is the minimum cut depth in its cut set plus one, as shown in Eq. (3). The node area of a node n, $A(n)$, is the cut area of the cut with the minimal depth in the cut set plus one, as shown in Eq. (4). For the PI nodes, their delay and area are 0.

$$D(c) = \max_{n\ \in\ cut} \{D(n)\} \tag{1}$$

$$A(c) = \sum_{n\ \in\ cut} \{A(n)\} \tag{2}$$

$$D(n) = \min_{c\ \in\ cut\ set_n} \{D(c)\}\ +\ 1 \tag{3}$$

$$A(n) = \min_{c\ \in\ cut\ set_n\ with\ min\{D(n)\}} \{A(c)\}\ +\ 1 \tag{4}$$

An example of cut enumeration process with K = 5 is shown in Fig. 6. The given AIG is illustrated in Fig. 6(a). The nodes a to f are PIs. The parentheses at the right of nodes represent the

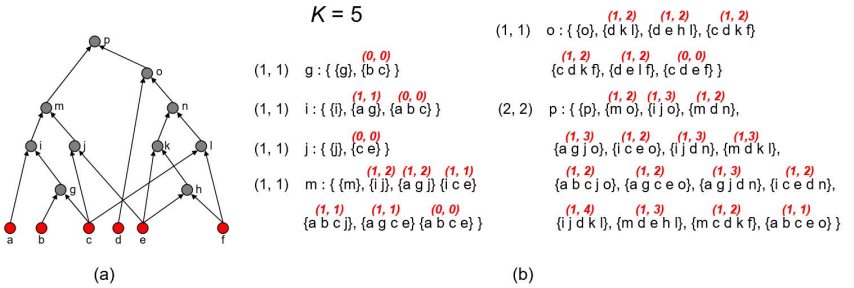

Fig. 6. An example of cut enumeration with label.

cut sets of nodes, and the inner parentheses in a parenthesis are the cuts in the cut set. To generate the cut of a node n, we put the node n itself into cut set first. Then, use the aforementioned generation function to generate the complete cut set. The labels on the top of inner parentheses represent the cut depth and cut area, $(D(c), A(c))$, and the labels at the left of the nodes represent the node depth and node area, $(D(n), A(n))$. When labeling the depth and area of a node, we consider the cut with the minimal cut depth in the set first. If there are more than one cut having the minimal cut depth in the set, we choose the cut with the minimal cut area. In Fig. 6(b), we can see that all the cuts in the cut set of node p are with the same depth. In this case, we choose the cut with the minimal $A(c)$ to calculate the depth and area of node p since it takes the minimal number of SABs. Therefore, we find that the cut $\{a, b, c, e, o\}$ is the best cut for node p due to its smallest cut area.

However, when circuits and K become larger, enumerating all K-feasible cuts is impractical. In our observation, many enumerated cuts are redundant, which means that the cuts are useless for mapping a node. Furthermore, these redundant cuts generate more redundant cuts. Therefore, we apply a priority cut heuristic [21] to reduce the number of enumerated cuts. The strategy is to reserve only a small fixed number p, which is typically five to ten, of "good" K-feasible cuts for each node. The "good" K-feasible cuts are the cuts that have smaller cut depths and the smaller cut areas in the cut set. However, there is a difference between the strategies of priority cut in [21] and this work. In this work, we keep at most p "good" cuts for each cut size from 2 to K. Moreover, the trivial cut of node n, $\{n\}$, which consists of the node itself, is always added to the cut set to ensure that any nodes can be mapped into an SAB. Then, we keep additional p cuts with the optimal cut depth plus one for the cuts whose cut size is K according to the two observations from our early experiments. In the first observation, the larger the cut size is, the more nodes are likely to be packed into an SAB. Therefore, we reserve additional p "good" cuts for the cuts with the cut size of K. In the second observation, the number of nodes in a cut is similar to other cuts with the same cut depth. That is, if a cut fails to map its function onto an SAB with a width W in the second phase, other cuts with the same depth are very likely to fail as well. Therefore, instead of keeping all the cuts with the optimal depth, we choose to retain the cuts with the optimal depth plus one. For example, if $p = 3$ and $K = 5$, we keep at most three cuts for each cut size from 2 to 4. For the cut size of 1, we only keep one cut, the trivial cut. For the cut size of 5, we keep three cuts with the optimal depth and three other cuts with the optimal depth plus one. The maximum number of the enumerated cuts in a cut set in our strategy is $K \times p+1$.

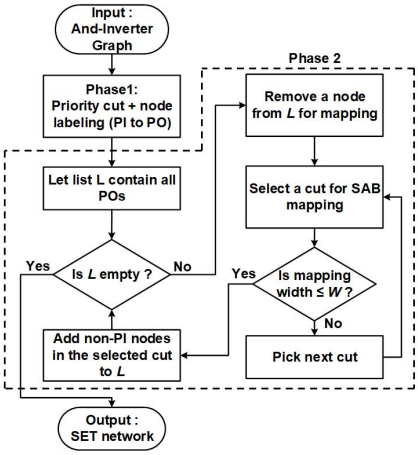

Fig. 7. Overall flow of the proposed synthesis approach.

B. Phase 2: SAB Mapping

In this phase, we transform the given Boolean circuit into an SET network consisting of (K, W)-SABs. The steps of SAB mapping are as follows: First, list all PO nodes in L. Second, choose one node from L for mapping, and sort the cuts of the chosen node by 1) cut depth, 2) cut area, and 3) cut size. We first select the cut with the optimal cut depth for mapping the node, therefore, we can have the mapped SAB with the minimal delay in the cut set. If there are more than one cut having the optimal cut depth, we choose the optimal-depth cut with the minimal cut area for mapping to reduce the number of SABs. When we have multiple cuts that have the optimal cut depth and the minimal cut area, we choose the cut with the largest cut size to pack more nodes in an SAB. Once we select a cut of the node for mapping, we apply the SET array synthesis method in [3] to obtain the mapped SAB. If the mapping width of the SAB is smaller than or equal to the predetermined W, the mapping is successful. However, if the mapping width of the SAB is larger than W, we continue to choose the next highest priority cut for mapping until the mapping width constraint is satisfied.

In this work, it is guaranteed to map a node onto the SAB with the width smaller than or equal to W successfully since

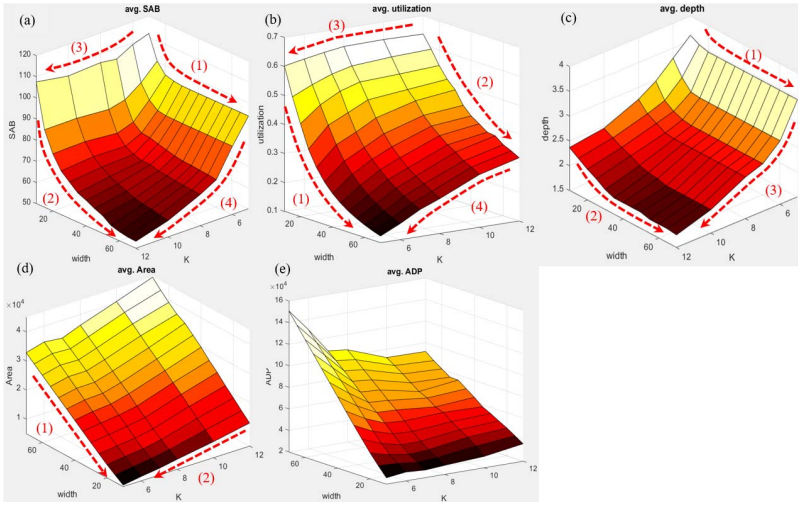

Fig. 8. Experimental Results. (a) Average number of required SABs. (b) Average utilization of SABs. (c) Average depth. (d) Average area. (e) Average Area Delay Product (ADP).

our priority cut strategy always keeps the trivial cut. Once a node is successfully mapped onto an SAB with the chosen cut, we add all the non-PI nodes, except for the nodes that have already been mapped, in the cut into the list L. The overall flow is shown in Fig. 7. The process is not terminated until L is empty. In the end, we obtain an SET network composed of mapped (K, W)-SABs.

IV. Experimental results

We implemented the proposed algorithm in C++ language and conducted experiments on an Intel Xeon 2.4GHz CPU platform with 64 GBytes memory. The experiments were conducted over a set of MCNC [24] and IWLS 2005 [28] benchmarks represented by AIG. We conducted experiments for different sizes of (K, W)-SABs, where K is varied from 5 to 12 and W is varied from 10 to 70 at intervals of 5. The averaged experimental results of this set of benchmarks are illustrated in Fig. 8, which was drawn with MATLAB.

Fig. 8(a) shows the average number of required SABs with different sizes. Fig. 8(b) shows the average utilization with different sizes of SABs. The utilization of an SAB refers to the ratio of the mapped width with respect to W. For example, if the mapped width of a sub-function on an SAB is 8, and the parameter W is 10, the utilization of the SAB is 80%. The line (1) in Fig. 8(a) indicates the effect of increasing width W to the average number of SABs with a smaller K of 5. In the beginning, we found that the average number of required SABs was greatly reduced when the W became larger. However, when the W became much larger, the average number of SABs is almost intact. It indicates that we do not need SABs with very large widths when K is small since only a few nodes are packed into an SAB. Furthermore, the line (1) in Fig. 8(b) also explains that the utilization dramatically dropped when the W became larger. The line (2) in Fig. 8(a) indicates that the number of required SABs with a larger K of 12 continuously decreases when W increases. In Fig. 8(b), the line (2) shows that the averaged utilization of SABs decreases more gently than the line (1) with the increase of W. Furthermore, the averaged utilization of SABs for a large W, which is indicated by the line (4), is lower than the line (3).

In Fig. 8(c), the lines (1) and (2) indicate that the averaged depth of SABs becomes smaller when the W becomes larger. This means that an SAB with a larger width can accommodate a larger sub-function. However, this is not obvious anymore when the width of SABs is too large. The line (3) also indicates that the averaged depth of SET networks becomes smaller when K increases. This is because more nodes are packed into an SAB with a larger K.

We observed that using SABs with larger parameters K and W leads to a smaller averaged number of required SABs for mapping in an SET network. However, the lines (1) and (2) in Fig. 8(d) indicate that the averaged mapping area of the SET network with larger size of SABs is oppositely larger. The minimum averaged area occurs when the parameter K is 5 and W is 10.

For the averaged Area Delay Product (ADP), which is calculated as $area \times depth$, with different sizes of SABs in Fig. 8(e) indicates the minimum ADP occurs when K is 5 and W is 10, and a similar ADP occurs when K is 7 and W is 10.

TABLE I shows the detailed experimental results with (5, 10) and (7, 10) SABs in two columns. The last row shows the averaged results among these benchmarks. It indicates that the averaged area of the SET networks with (5, 10) SABs is smaller than the SET networks with (7, 10) but the averaged depth of the SET networks with (5, 10) SABs is larger than the SET networks with (7, 10). Therefore, the two averaged ADP of the two experiments with (5, 10) and (7, 10) SABs are similar.

V. Conclusion

In this paper, we propose the first delay minimization synthesis algorithm decomposing and mapping a Boolean circuit into a set of fixed size (K, W)-SABs. The proposed method consists of two phases that guarantee the mapped SET SABs meet the predefined height and width. The experimental results suggest that smaller parameters K and W lead to

TABLE I
The experimental results with (K, W) = (5, 10), and (K, W) = (7, 10) SABs.

Bench.	$K = 5$, $W = 10$						$K = 7$, $W = 10$					
	|SAB|	utiliz. (%)	depth	area	ADP	T(s)	|SAB|	utiliz. (%)	depth	area	ADP	T(s)
alu2	184	63.1	9	9200	82800	0.94	159	63.6	6	11130	66780	2.74
alu4	345	64.6	10	17250	172500	1.95	333	65.9	7	23310	163170	5.75
apex6	291	63.8	5	14550	72750	1.28	256	67.1	4	17920	71680	2.69
apex7	74	59.5	4	3700	14800	0.27	66	65.2	3	4620	13860	0.57
b9	42	56.7	3	2100	6300	0.16	38	61.1	3	2660	7980	0.29
c8	38	65.5	3	1900	5700	0.20	34	62.4	3	2380	7140	0.31
c17	3	56.7	2	150	300	0.02	3	56.7	2	210	420	$<$0.01
cc	23	56.1	2	1150	2300	0.09	23	47.8	2	1610	3220	0.17
cht	45	72.9	2	2250	4500	0.17	38	77.4	2	2660	5320	0.12
cm85	16	56.2	3	800	2400	0.07	17	57.1	2	1190	2380	0.12
cm138	11	71.8	2	550	1100	0.09	11	71.8	2	770	1540	0.14
cm151	8	70.0	3	400	1200	0.04	9	63.3	2	630	1260	0.13
cm162	14	65.0	3	700	2100	0.05	14	54.3	2	980	1960	0.15
cm163	13	56.2	2	650	1300	0.07	13	72.3	2	910	1820	0.14
cmb	19	54.2	3	950	2850	0.06	16	60.0	3	1120	3360	0.14
count	43	67.2	5	2150	10750	0.29	44	73.9	4	3080	12320	0.57
cu	19	52.6	3	950	2850	0.05	17	55.3	2	1190	2380	0.06
example2	121	53.9	3	6050	18150	0.53	115	59.6	3	8050	24150	1.49
frg1	29	64.0	5	1450	7250	0.11	31	57.1	4	2170	8680	0.25
frg2	312	65.2	4	15600	62400	1.37	294	68.6	4	20580	82320	4.53
i1	20	44.0	3	1000	3000	0.05	17	45.3	2	1190	2380	0.04
i2c	402	59.5	5	20100	100500	1.75	375	65.9	4	26250	100500	3.65
i8	380	71.0	4	19000	76000	2.68	398	77.3	3	27860	83580	11.12
lal	35	54.6	3	1750	5250	0.19	32	60.0	3	3350	6720	0.46
ldd	42	59.0	3	2100	6300	0.31	39	58.7	2	2730	5460	0.47
pcie	23	57.7	3	1150	3450	0.10	16	70.6	2	1120	2240	0.11
pcier8	37	55.9	4	1850	7400	0.15	34	60.3	3	2380	7140	0.47
pm1	20	47.5	2	1000	2000	0.06	19	51.6	2	1330	2660	0.11
Avg.	93.2	60.2	3.7	4658.9	24221.4	0.47	87.9	62.5	3.0	6192.1	24729.3	1.31

smaller ADP. Furthermore, we also showed that the proposed priority cut strategy makes our algorithm very efficient.

REFERENCES

[1] N. Asahi *et al.*, "Single-electron logic device based on the binary decision diagram," *IEEE Trans. Elec. Dev.*, 1997.

[2] R. E. Bryant, "Graph-based algorithms for Boolean function manipulation," *IEEE Trans. Computers*, 1986.

[3] Y.-H. Chen *et al.*, "Area minimization synthesis for reconfigurable single-electron transistor arrays with fabrication constraints," *ACM J. Emerg. Tech. Com. Syst.*, 2016.

[4] Y.-H. Chen *et al.*, "ROBDD-based area minimization synthesis for reconfigurable single-electron transistor arrays," *in Proc. Int. Symp. VLSI Design, Auto. Test*, 2015.

[5] Y.-C. Chen *et al.*, "Automated Mapping for Reconfigurable Single-Electron Transistor Arrays", *in Proc. Design Auto. Conference*, 2011.

[6] Y.-C. Chen *et al.*, "A Synthesis Algorithm for Recongurable Single-electron Transistor Arrays," *ACM J. Emer. Tech. Comp. Syst.*, 2013.

[7] Y.-C. Chen *et al.*, "Verification of Reconfigurable Binary Decision Diagram-based Single-Electron Transistor Arrays," *IEEE Trans. CAD of Int. Circuits and Syst.*, 2013.

[8] C.-E. Chiang *et al.*, "On reconfigurable single-electron transistor arrays synthesis using reordering techniques," *in Proc. Des. Auto. and Test in Europe*, 2013.

[9] J. Cong *et al.*, "Cut ranking and pruning: Enabling a general and efficient FPGA mapping solution." *in Proc. ACM/SIGDA S. Int. S. FPGA*, 1999.

[10] S. Eachempati *et al.*, "Reconfigurable Bdd-based Quantum Circuits," *in Proc. Int. Symp. Nanosc. Archit.*, 2008.

[11] H. Hasegawa *et al.*, "Hexagonal binary decision diagram quantum logic circuits using Schottky in-plane and wrap gate control of GaAs and InGaAs nanowires," *Phys. E, Low-dimensional Syst. Nanostruct.*, 2001.

[12] C.-H. Ho *et al.*, "Area-aware decomposition for single-electron transistor arrays," *ACM Trans. Des. Auto. Elec. Syst.*, 2016.

[13] C.-Y. Huang *et al.*, "A Defect-aware Approach for Mapping Reconfigurable Single-Electron Transistor Arrays," *in Proc. Asia and South Pacific Design Automation Conference.*, 2015.

[14] C.-Y. Huang *et al.*, "Diagnosis and Synthesis for Defective Reconfigurable Single-Electron Transistor Arrays," *IEEE Trans. VLSI Syst.*, 2016.

[15] S. Kasai *et al.*, "A single electron binary-decision-diagram quantum logic circuit based on Schottky wrap gate control of a GaAs nanowire hexagon," *Elec. Device Lett.*, 2002.

[16] S. Kasai *et al.*, "Fabrication of GaAs-based integrated 2-bit half and full adders by novel hexagonal BDD quantum circuit approach," *in Proc. Int. Symp. Semi. Dev. Res.*, 2001.

[17] Y.-J. Li *et al.*, "Dynamic Diagnosis for Defective Reconfigurable Single-Electron Transistor Arrays," *IEEE Trans. VLSI Syst.*, 2017.

[18] C.-W. Liu *et al.*, "Width Minimization in the Single-Electron Transistor Array Synthesis," *in Proc. Des., Auto. and Test in Europe*, 2014.

[19] C.-W. Liu *et al.*, "Synthesis for width minimization in the single-electron transistor array," *IEEE Trans. VLSI Syst.*, 2015.

[20] L. Liu *et al.*, "A reconfigurable low-power BDD logic architecture using ferroelectric single-electron transistors," *IEEE Trans. Elec. Device*, 2015.

[21] A. Mishchenko *et al.*, "Combinational and sequential mapping with priority cuts," *IEEE Trans. Computer-Aided Design*, 2007.

[22] H. W. Ch. Postma *et al.*, "Carbon nanotube single-electron transistors at room temperature," *Science*, 2001.

[23] Y.-T. Tan *et al.*, "Room temperature nanocrystalline silicon single-electron transistors," *J. Appl. Phys.*, 2003.

[24] S. Yang, "Logic Synthesis and Optimization Benchmarks, Version 3.0," *Tech. Report, Microelectronics Center of North Carolina*, 1991.

[25] Z. Zhao *et al.*, "BDD-Based Synthesis of Reconfigurable Single-Electron Transistor Array," *in Proc. Int. Conference CAD*, 2014.

[26] L. Zhuang *et al.*, "Silicon single-electron quantum-dot transistor switch operating at room temperature," *Appl. Phys. Lett.*, 1998.

[27] *Int. Tech. Road. Semi., Semiconduc. Industry Association*, 2006.

[28] *IWLS 2005 Benchmarks. (June 2005). Retrieved March, 2015* [Online]. Available: http://iwls.org/iwls2005/benchmarks.htm