

A Defect-aware Approach for Mapping Reconfigurable Single-Electron Transistor Arrays

Ching-Yi Huang
Computer Science
National Tsing Hua University
Hsinchu, Taiwan, R.O.C.
s9862516@m98.nthu.edu.tw

Chian-Wei Liu
Computer Science
National Tsing Hua University
Hsinchu, Taiwan, R.O.C.
smilelcw@gmail.com

Chun-Yao Wang
Computer Science
National Tsing Hua University
Hsinchu, Taiwan, R.O.C.
wcyao@cs.nthu.edu.tw

Yung-Chih Chen
Computer Science & Engineering
Yuan Ze University
Chung Li, Taiwan, R.O.C.
ycchen.cse@saturn.yzu.edu.tw

Suman Datta
Electrical Engineering
Pennsylvania State University
University Park, PA, U.S.
sdatta@enr.psu.edu

Vijaykrishnan Narayanan
Computer Science & Engineering
Pennsylvania State University
University Park, PA, U.S.
vijay@cse.psu.edu

Abstract—Single-Electron Transistor (SET) at room temperature has been demonstrated as a promising device for extending Moore’s law due to its ultra low power consumption. However, early realizations of SET array lacked variability and reliability due to their fixed architectures and high defect rates of nanowire segments. Therefore, a reconfigurable version of SET was proposed to deal with these issues. Recently, several automated mapping approaches were proposed for area minimization of reconfigurable SET arrays. However, to the best of our knowledge, no mapping approaches that consider the existence of defective nanowire segments were proposed. Thus, this paper presents the first defect-aware approach for mapping reconfigurable SET arrays. The experimental results show that our approach can successfully map the SET arrays with 20% width overhead on average in the presence of 5000 ppm defects.

I. INTRODUCTION

Power consumption has become one of the primary bottlenecks to meet the Moore’s law. To deal with this issue, many emerging low power devices have been explored in recent years. Among these devices, several demonstrations of the room temperature operation of Single-Electron Transistors (SETs) have proved that these devices are promising candidates that substitute traditional Complementary Metal-Oxide-Semiconductor (CMOS) devices for future designs [3] [15] [16].

Since only a few electrons are involved in the switching process, SETs suffer from low transconductance. Therefore, the conduction mechanism of the conventional CMOS-based logic is not applicable to SETs. To this end, a binary decision diagram (BDD)-based [2] architecture was proposed as a feasible platform for implementing logic functions using SETs [1]. Using this, a Boolean circuit can be implemented through

mapping the BDD of the Boolean function onto an SET array, which is represented as a hexagonal nanowire network controlled by Schottky wrap gates [10][11].

However, the realization of the architecture proposed in [11] is fixed and not amenable to functional reconfiguration. Furthermore, if any of the nanowire segments or the wrap gates is defective, the whole circuit becomes useless. This causes its low yield due to a high defect rate of nanowires and nanodevices. Fortunately, a reconfigurable version of SET using wrap gate tunable tunnel barriers was proposed [9] to increase the flexibility and reliability of the SET arrays [1][10][11]. The electrostatic properties through in-depth device simulation were also presented in [12][14], which showed that this device can provide an energy-delay product that is an order of magnitude lower than traditional CMOS devices.

Fig. 1(a) illustrates the structure of the reconfigurable SET array, which can be divided into three layers. The bottom layer is the device layer, which is composed of the hexagonal nanowire network with wrap gates. The middle layer is used to configure the operating modes of every transistor. The top layer takes charge of input signal connections to SETs. If some nanowires or wrap gates (bottom layer) are defective after manufacturing, we still have a usable SET array through configuration in the middle layer.

Recently, this success of SET array realization attracts the development of its automation tools [4]-[8][13][17]. The first automated synthesis approach was proposed in [4], which presented a product-term-based approach that synthesizes a logic circuit by mapping all its product terms into the SET architecture. Based on the product-term-based approach, [6][8] proposed mapping approaches to reduce the number of hexagons of the mapped SET arrays under *symmetric fabric constraint* by using *product term reordering* and *variable reordering* techniques. Additionally, [8] also tried to form *ladder-like* and *branch-than-share* mapping structures to reduce the number of hexagons needed. However, the area of an SET array on a chip is the product of its bounded height and width. As a

This work is supported in part by the Ministry of Science and Technology of Taiwan under Grant MOST 103-2221-E-007-125-MY3, MOST 103-2221-E-155-069, NSC 102-2221-E-007-140-MY3, and NSC 100-2628-E-007-031-MY3.

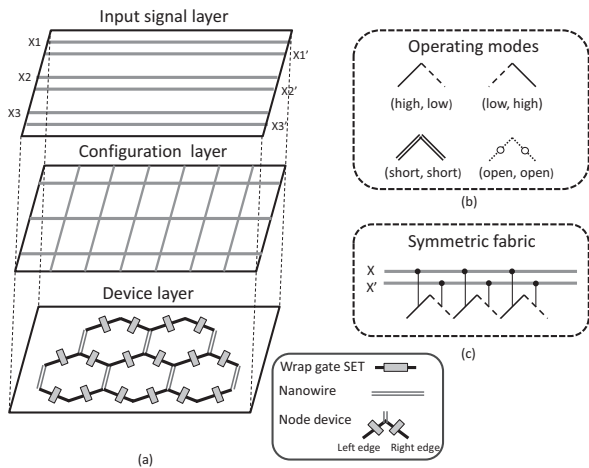


Fig. 1. (a) Physical architecture of a reconfigurable SET array. (b) Operating modes. (c) Symmetric fabric architecture for input wiring.

result, [7][13] proposed width minimization approaches¹. [7] dynamically chose the best unmapped product terms to achieve a maximal sharing. [13] focused on minimizing the number of product terms and fully utilizing the architecture flexibility under the symmetric fabric constraint to obtain more compact SET arrays. In parallel with the development of mapping techniques, the first satisfiability (SAT)-based verification method for SET arrays was also proposed [5].

Although the previous mapping techniques are effective and efficient, they all assume that the device layer of SET array is defect-free. This assumption, however, does not match the real situation. Therefore, if some nanowires or wrap gates are defective after manufacturing, previous approaches might fail to map. This is because the mapping process must avoid the defective areas for correct mapping. As a result, in this paper, we propose a defect-aware approach when mapping SET arrays.

In this work, we also propose a defect model, which considers three types of failures: *single-stuck-at-open*, *double-stuck-at-open*, and *stuck-at-short*, on SET arrays. We then propose a baseline defect-aware mapping algorithm, which can successfully map SET arrays by detouring the defective devices. Finally, we further propose a defect-reuse mapping algorithm that reuses defective device for width reduction.

We conducted an experiment in this work. The IWLS2005 benchmarks [18] are the circuits to be mapped in the experiments. In the experiment, we uniformly and randomly injected 0.5% defects into SET arrays. This defect rate is a user defined parameter. However, 0.5% or 5000 *ppm* is higher than that in conventional CMOS process [19]. The experimental results show that our approach can successfully map the SET array. The width increases are 20.48% and 16.42% on average in the baseline algorithm and in the defect-reuse algorithm, respectively, compared to the defect-free SET arrays.

The rest of the paper is organized as follows. Section II introduces the backgrounds of this work. Section III presents the defect model and the proposed defect-aware mapping algorithms. The experimental results are shown in Section IV.

¹The height of an SET array is usually the same with the number of inputs in a circuit.

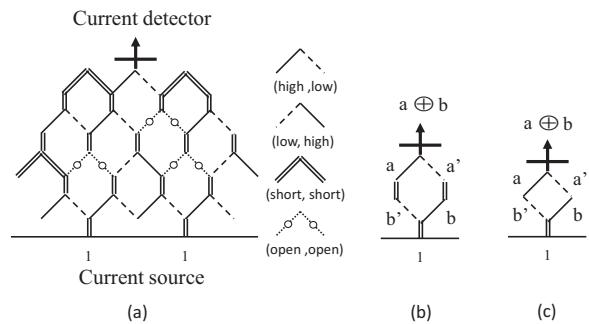


Fig. 2. (a) An SET array. (b) An example of $a \oplus b$. (c) A simplified diamond-shaped network of $a \oplus b$ [4].

Section V concludes this paper and presents future works.

II. PRELIMINARIES

A. Reconfigurable SET Array

A reconfigurable SET array can be represented as a hexagonal network as shown in Fig. 2(a). At the top of the SET array, there is a current detector measuring the current coming from the current source, represented as 1, at the bottom. When the electrons transported from the current sources are detected by the current detector through a conducting path, which is controlled by the input variables, the output value of the Boolean circuit is 1; otherwise, it is 0. Each sloping edge in the SET array can be configured as one of four modes: *high*, *low*, *short*, or *open*. A *high* (*low*) edge indicates that the corresponding SET device operates in active mode controlled by a variable x (x'). Furthermore, a *short* (*open*) edge is electrical *short* (*open*), where the corresponding SET device operates in *short* (*open*) mode. For example, Fig. 2(b) shows an implementation of $a \oplus b$. The current detector detects the current and the function will be evaluated as 1 when either $(a = 1, b = 0)$ (left path) or $(a = 0, b = 1)$ (right path).

Since all the vertical edges of the hexagons are electrically *short*, for ease of discussion, only the sloping edges are preserved in the abstract graph. Fig. 2(c) is the corresponding diamond-shaped network of the hexagonal network in Fig. 2(b). Note that the connections to the current source are *short* edges as well.

B. Symmetric Fabric Constraint

To reduce the routing area of input wiring in SET arrays, *symmetric fabric constraint* is imposed in all the previous works. Symmetric fabric constraint enforces that a pair of left edge and right edge of a *node device* must be one of (*high*, *low*), (*low*, *high*), (*short*, *short*), or (*open*, *open*), as shown in Fig. 1(b), and both (*high*, *low*) and (*low*, *high*) configurations are not allowed to appear in the same row simultaneously, as shown in Fig. 1(c).

C. Branch-then-Share

Branch-then-Share product terms are two product terms that branch in one row and merge in the succeeding row such

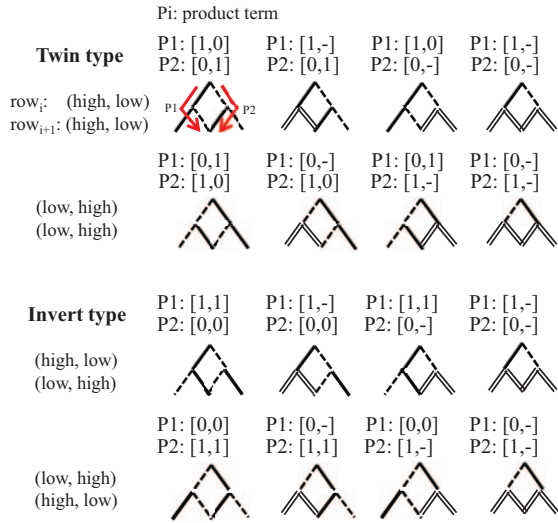


Fig. 3. All types of Branch-then-Shares [13].

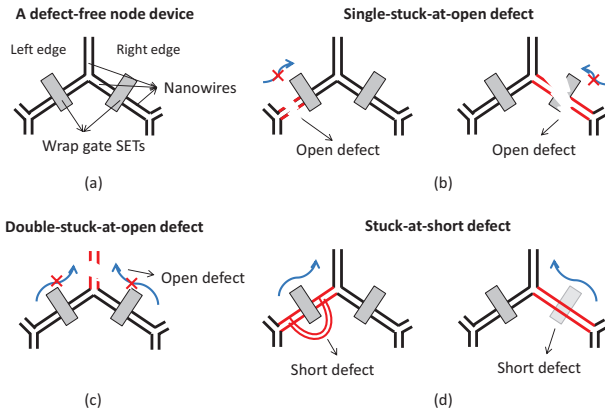


Fig. 4. The proposed defect model. (a) A defect-free node device. (b) Examples of single-stuck-at-open defects. (c) An example of a double-stuck-at-open defect. (d) Examples of stuck-at-short defects.

that the remaining edges are all shared [8][13]. These structures effectively shrink the width of mapping area. In this paper, we also consider all types of *two-bit Branch-then-Share*, as shown in Fig. 3 [13], in our mapping approach.

III. DEFECT-AWARE MAPPING ALGORITHM

In this section, we first introduce the proposed defect model. Based on this defect model, we then propose a baseline defect-aware mapping algorithm, which can successfully map SET arrays by detouring the defective devices. Finally, we further propose a defect-reuse mapping algorithm that reuses defective devices for width reduction.

A. Defect Model

Open and *short* are common types of failures that occur in the SET devices. These failures can occur when the programmable gate of the SET device is defective and results in either a permanent short or open conditions. Short could arise

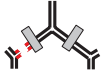
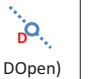

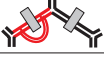
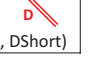
Failure types	Representation
Single-stuck-at-open:	 or  (DOpen, -) (-, DOpen)
Double-stuck-at-open:	 (DOpen, DOpen)
Stuck-at-short:	 or  (DShort, -) (-, DShort)

Fig. 5. The fabric representations of the three types of failures.

from variation in fin thickness leading to lack of pinch off by depletion region under the split gate. Open could arise from excessive thinning of the etched fin leading to physical disruption in the continuity of the branch. In addition, the nanowires connected to the SET devices would also suffer from open and short failures caused by manufacturing defects. Fig. 4(a) shows a defect-free node device. If an open defect occurs on a SET device or occurs on a nanowire connected to the SET device, i.e., on the left edge or the right edge of a node device as shown in Fig. 4(b), the electrons transported from the lower node device will never pass through the wrap gate. This single defective edge is named as a *single-stuck-at-open* defect. If an open defect occurs on a vertical nanowire of a hexagon, as shown in Fig. 4(c), the electrons transported from the lower SETs will never pass through the vertical edge. In this case, we say that this node device has a *double-stuck-at-open* defect. If a short defect occurs on a SET device or occurs on the nanowires connected to a SET device, as shown in Fig. 4(d), this defective edge will be always conducted. We name this defect a *stuck-at-short* defect.

In the succeeding discussion, these three types of failures are considered. Their corresponding representations on the network are shown in Fig. 5, where *DOpen* and *DShort* represent the permanent configurations caused by the defects and their functionalities are the same as *open* and *short* configurations, respectively.

B. Baseline Defect-aware Mapping

The previous works [6][13] focused on developing reordering techniques and mapping approaches for minimizing the area of defect-free SET arrays. In this work, we adopt the result of product terms after the reordering, and focus on the mapping algorithm for a defect-aware version.

When defects occur in SET arrays, we have to deal with them in the mapping process. Since defects are considered *open* or *short*, they can be treated as certain already been configured edges that might be against the symmetric fabric constraint. In the mapping process, for open defects, they might be obstacles for conducting a path. For short defects, however, they might be pitfalls resulting in invalid paths.

In the baseline defect-aware mapping algorithm, if there is only one *DOpen* edge or *DShort* edge at a node device, we configure the other edge as *open* before mapping. Additionally, since the *DShort* edge possibly causes undesired invalid paths,

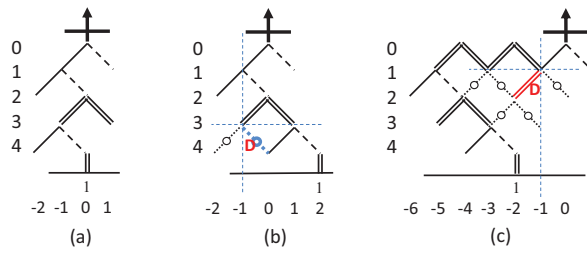


Fig. 6. The examples of baseline defect-aware mapping algorithm. (a) The mapping result of $10 - 0$ without defects. (b) The mapping result when there is a $DOpen$ at the right edge of the node at $(-1, 3)$. (c) The mapping result when there is a $DShort$ at the left edge of the node at $(-1, 1)$.

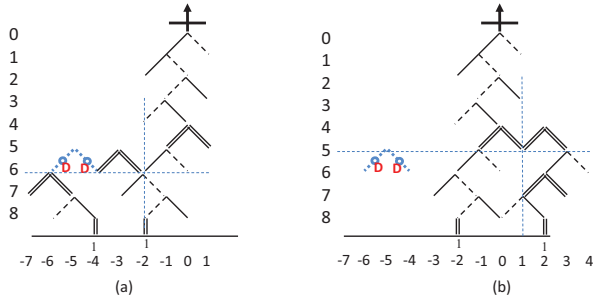


Fig. 7. (a) An example of expansion meeting defects. (b) An example of modified expansion.

we configure (*open, open*) to node devices that are adjacent to and below the $DShort$ edge for isolation. We call these (*open, open*) configurations a *side protection* and a *bottom protection*, respectively, in this paper.

For example, Fig. 6(a) shows the original mapping result of the product term $10 - 0$. If there occurs a $DOpen$ defect at the right edge of the node at $(x, y) = (-1, 3)$ in the SET array, as shown in Fig. 6(b), we detour the mapping and add a (*high, low*) configuration at $(x, y) = (1, 3)$ for mapping the product term. If there occurs a $DShort$ defect at the left edge of the node at $(-1, 1)$, as shown in Fig. 6(c), we add a *side protection* and a *bottom protection* at $(-3, 1)$ and $(-2, 2)$, respectively, before mapping the product term. Then, we expand two nodes to the left at the first row to detour the defect, and the (*high, low*) configuration at $(-1, 1)$ of Fig. 6(a) is moved to $(-5, 1)$. The rest of mapping follows the ordinary mapping algorithm.

In the last examples, we use *short* edges to change the configuration direction or use expansion operation (*short* edges as well) to detour the defective areas for successful mapping. However, the expansion operation itself might encounter defective areas as well. If the expansion meets defective areas, the mapping process looks for another edge at upper rows for a trial of expansion again. For example, as shown in Fig. 7(a), assume that the product term $1001-100$ has been mapped. We would like to map the product term $1001-1-1$, and an expansion at $(-2, 6)$ is required. However, during the expansion to the left for two nodes, we meet a ($DOpen, DOpen$) defect. Therefore, we discard this expansion and expand at another location of $(1, 5)$ as shown in Fig. 7(b). Finally, the product term $1001-1-1$ is successfully mapped.

Note that the connections to the current sources could be de-

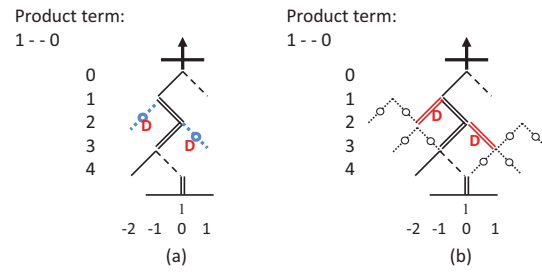


Fig. 8. Examples demonstrating that $DOpen$ and $DShort$ defects are not on the path to be configured.

fective and is also considered in this paper. However, to ensure that the expansion can be performed eventually for successful mapping, we assume that no $DOpen$ edges occur at the first row of SET array. Otherwise, the mapping process would fail with a high probability. Furthermore, we also assume that no defect occurs at $(0, 0)$ since $(0, 0)$ is the location connecting to the current detector. Any defect occurring at $(0, 0)$ would directly make the mapping process fail.

C. Defect-reuse Mapping

Since the defects are *open* or *short*, and they are the same with normal configurations, except singular (one edge) configuration, we can reuse the defects if they are applicable for width reduction. Therefore, in this subsection, we propose a defect-reuse mapping approach.

For a single $DOpen$ or $DShort$ defect occurring at a node device, we leave the other edge intact such that it can be configured as any desired type. Therefore, if the defective edges are not on the conducting path during mapping a product term, we can configure the other edge of the defective node device as usual. For example, Figs. 8(a) and 8(b) show valid configurations when $DOpen$ or $DShort$ defects are not on the path to be configured. We reuse the other edge of a defective node to form the conducting paths.

In the mentioned defect-aware approach, we isolate $DShort$ edges using *side protection* and *bottom protection* to avoid creating invalid conducting paths. However, $DShort$ edges could be reused when *short* configurations are needed. Therefore, we do not isolate $DShort$ edges at once. Instead, we dynamically add protection for $DShort$ edges. Specifically, if we map a product term downward, we add the *side protection* of the $DShort$ edges only, and remove the *bottom protection*. If we expand the mapping to the left or right, we add the *bottom protection* of the $DShort$ edges and remove the *side protection*. For example, assume that there are two $DShort$ s in the SET array of Fig. 9(a). The original *side protection* and *bottom protection* are shown in Fig. 9(a) as well. When mapping a product term $100-0100$, however, we remove the *bottom protection* at $(0, 4)$. As a result, the $DShort$ edge at $(-1, 3)$ can be reused for the don't care bit of the product term as shown in Fig. 9(b). For mapping another product term $100-01-1$, we need to expand to left at $(-2, 6)$. Therefore, we remove the *side protection* at $(-5, 5)$ such that the $DShort$ edge at $(-3, 5)$ can be reused as well. Fig. 9(c) shows the mapping results reusing the $DShort$ edges.

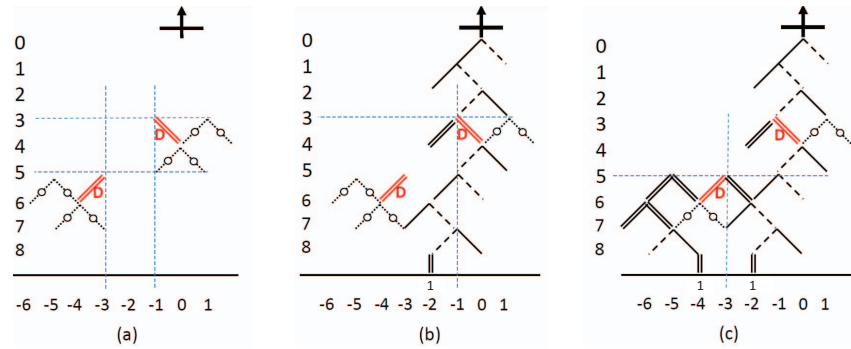


Fig. 9. (a) Two *DShort* edges occur in an SET array. (b) The mapping result after reusing the *DShort* at (-1, 3). (c) The mapping result after reusing both the *DShorts* at (-1, 3) and (-3, 5).

IV. EXPERIMENTAL RESULTS

We implemented the proposed algorithms in C language. The experiment was conducted over a set of IWLS 2005 benchmarks [18] on a 3.0 GHz Linux platform. We re-implemented the product term extraction technique proposed in [13] for obtaining the product terms. Then these product terms were analyzed and reordered through the re-implemented reordering and Branch-then-Share collection techniques proposed in [13]. The experiment shows the result comparison between different mapping algorithms.

In the experiment, we uniformly and randomly injected the defects into the SET array. The defect rate was set as 0.5% of the number of sloping edges of the SET array where 0.1% is for stuck-at-short defects, 0.2% is for single-stuck-at-open defects, and 0.2% is for double-stuck-at-open defects. We conducted the mapping algorithms for 10 defect maps of a benchmark to obtain the average results.

Table I summarizes the experimental results of our experiment. Columns 1-3 list the benchmarks and the number of primary inputs and primary outputs. Columns 4 and 5 list the width of the mapping result and the CPU time for defect-free SET arrays, respectively. Columns 6-8 show the average width of the mapping result, the ratio of the increased width compared to defect-free mapping, and the CPU time in the baseline defect-aware algorithm. Columns 9-11 show the corresponding results in the defect-reuse algorithm.

For example, the benchmark *pcler8* has 27 primary inputs and 17 primary outputs. The original mapping algorithm requires 0.11 seconds to map the benchmark on a defect-free SET array with 256 width. The baseline defect-aware mapping algorithm requires 0.11 seconds to obtain the average result of 287 width, or 12.11% $((287 - 256)/256 \times 100\%)$ increased width compared to the original width. The defect-reuse algorithm requires 0.16 seconds to map the SET array with 8.91% increased width compared to the original width.

According to Table I, we can see that the widths of the mapping results obtained by the defect-reuse mapping algorithm are less than that by the baseline defect-aware algorithm with a little CPU time suffering. On average, the ratios of increased width are 20.48% and 16.42% in the baseline defect-aware algorithm and in the defect-reuse algorithm, respectively. The experimental results show that the defect-reuse algorithm reduces the width growth.

V. CONCLUSION AND FUTURE WORK

In this paper, we propose the first defect-aware mapping approach for reconfigurable SET arrays. The proposed algorithms can successfully map SET arrays in the presence of defects. The experimental results show that the defect-reuse algorithm reduces more mapped width compared to the baseline algorithm. Our future work is to discuss the influence of reordering techniques on the defect-reuse mapping results.

REFERENCES

- [1] N. Asahi, M. Akazawa, and Y. Amemiya, "Single-Electron Logic Device Based on the Binary Decision Diagram," *IEEE Trans. Electron Devices*, vol. 44, pp. 1109-1116, July, 1997.
- [2] R. Bryant, "Graph-based Algorithms for Boolean Function Manipulation," *IEEE TC*, vol. 35, pp. 677-691, Aug. 1986.
- [3] H. W. Ch. Postma, T. Teepen, Z. Yao, M. Grifoni, and C. Dekker, "Carbon Nanotube Single-Electron Transistors at Room Temperature," *Science*, vol. 293, pp. 76-79, 2001.
- [4] Y.-C. Chen, S. Eachempati, C.-Y. Wang, S. Datta, Y. Xie, and V. Narayanan, "Automated Mapping for Reconfigurable Single-Electron Transistor Arrays," in *Proc. DAC*, pp. 878-883, 2011.
- [5] Y.-C. Chen, C.-Y. Wang, and C.-Y. Huang, "Verification of Reconfigurable Binary Decision Diagram-based Single-Electron Transistor Arrays," *IEEE TCAD*, pp. 1473-1483, 2013.
- [6] Y.-C. Chen, S. Eachempati, C.-Y. Wang, S. Datta, Y. Xie, and V. Narayanan, "A Synthesis Algorithm for Reconfigurable Single-Electron Transistor Arrays," *ACM JETC*, vol. 9, No. 1, Article 5, Feb. 2013.
- [7] Y.-H. Chen, J.-Y. Chen, and J.-D. Huang, "Area Minimization Synthesis for Reconfigurable Single-Electron Transistor Arrays with Fabrication Constraints," in *Proc. DATE*, 2014.
- [8] C.-E. Chiang, L.-F. Tang, C.-Y. Wang, C.-Y. Huang, Y.-C. Chen, S. Datta, and V. Narayanan, "On Reconfigurable Single-Electron Transistor Arrays Synthesis Using Reordering Techniques," in *Proc. DATE*, pp. 1807-1812, 2013.
- [9] S. Eachempati, V. Saripalli, V. Narayanan, and S. Datta, "Reconfigurable Bdd-based Quantum Circuits," in *Proc. Int. Symp. on Nanoscale Architectures*, pp. 61-67, 2008.
- [10] H. Hasegawa and S. Kasai, "Hexagonal Binary Decision Diagram Quantum Logic Circuits Using Schottky In-Plane and Wrap Gate Control of GaAs and InGaAs Nanowires," *Physica E: Low-dimensional Systems and Nanostructures*, vol. 11, pp. 149-154, 2001.

TABLE I
THE EXPERIMENTAL RESULTS OF OUR EXPERIMENT.

Benchmarks	PI	PO	Defect-free		Baseline			Defect-reuse		
			W_{ori}	$T(s)$	W_b	$IW(\%)$	$T(s)$	W_{dr}	$IW(\%)$	$T(s)$
C17	5	2	26	0.08	27.2	4.62	0.09	26.3	1.15	0.09
cm138a	6	8	64	0.09	98.4	53.75	0.09	93.7	46.41	0.09
x2	10	7	93	0.11	106.8	14.84	0.11	105.2	13.12	0.10
cm85a	11	3	180	0.09	187.3	4.06	0.10	186.4	3.56	0.10
cm151a	12	2	102	0.09	107.8	5.69	0.10	105.2	3.14	0.10
cm162a	14	5	134	0.10	142.0	5.97	0.09	139.2	3.88	0.10
cu	14	11	74	0.14	80.4	8.65	0.14	78.5	6.08	0.13
cm163a	16	5	71	0.10	85.1	19.86	0.10	81.4	14.65	0.09
cmb	16	4	35	0.10	49.8	42.29	0.10	49.4	41.14	0.10
pm1	16	13	81	0.10	94.7	16.91	0.10	91.8	13.33	0.10
pcl	19	9	89	0.10	124.9	40.34	0.09	121.3	36.29	0.10
sct	19	15	529	0.12	565.1	6.82	0.12	552.4	4.42	0.15
cc	21	20	137	0.10	157.0	14.60	0.10	148.7	8.54	0.10
il	25	16	76	0.10	98.2	29.21	0.11	90.4	18.95	0.11
lal	26	19	710	0.13	741.8	4.48	0.13	728.0	2.54	0.16
pcler8	27	17	256	0.11	287.0	12.11	0.11	278.8	8.91	0.12
c8	28	18	224	0.25	285.9	27.63	0.24	271.1	21.03	0.25
stepper.	29	29	2947	0.34	3170.9	7.60	0.42	3098.5	5.14	1.09
count	35	16	358	0.14	646.3	80.53	0.15	597.2	66.82	0.16
unreg	36	16	193	0.11	202.9	5.13	0.11	198.8	3.01	0.10
b9	41	21	795	0.15	868.8	9.28	0.19	848.7	6.75	0.25
cht	47	36	313	0.12	360.5	15.18	0.13	351.0	12.14	0.13
example2	85	66	1425	0.65	1686.9	18.38	0.66	1638.8	15.00	0.76
usb_phy	113	116	1292	0.33	1724.4	33.47	0.36	1642.1	27.10	0.43
sasc	133	129	2708	0.64	3537.8	30.64	0.67	3452.6	27.50	0.87
Average	-	-	-	0.18	-	20.48	0.18	-	16.42	0.23

- [11] S. Kasai, M. Yumoto, and H. Hasegawa, "Fabrication of GaAs-based Integrated 2-bit Half and Full Adders by Novel Hexagonal BDD Quantum Circuit Approach," in *Proc. Int. Symp. on Semiconductor Device Research*, pp. 622-625, 2001.
- [12] L. Liu, V. Saripalli, E. Hwang, V. Narayanan, and S. Datta, "Multi-Gate Modulation Doped In_{0.7}Ga_{0.3}As Quantum Well FET for Ultra Low Power Digital Logic," *Electro Chemical Society Transactions*, vol. 35, issue 3, pp. 311-317, 2011.
- [13] C.-W. Liu, C.-E. Chiang, C.-Y. Huang, C.-Y. Wang, Y.-C. Chen, S. Datta, and V. Narayanan, "Width Minimization in the Single-Electron Transistor Array Synthesis," in *Proc. DATE*, 2014.
- [14] V. Saripalli, L. Liu, S. Datta, and V. Narayanan, "Energy-Delay Performance of Nanoscale Transistors Exhibiting Single Electron Behavior and Associated Logic Circuits," *Journal of Low Power Electronics*, vol. 6, pp. 415-428, 2010.
- [15] Y. T. Tan, T. Kamiya, Z. A. K. Durrani, and H. Ahmed, "Room Temperature Nanocrystalline Silicon Single-Electron Transistors," *Journal of Applied Physics*, vol. 94, pp. 633-637, 2003.
- [16] L. Zhuang, L. Guo, and S. Y. Chou, "Silicon Single-Electron Quantum-Dot Transistor Switch Operating at Room Temperature," *Applied Physics Letters*, pp. 1205-1207, 1998.
- [17] Z. Zhao, C.-W. Liu, C.-Y. Wang, and W. Qian, "BDD-Based Synthesis of Reconfigurable Single-Electron Transistor Array," in *Proc. ICCAD*, pp. 47-54, 2014.
- [18] <http://iwls.org/iwls2005/benchmarks.html>
- [19] <http://www.xmultiple.com/xwebsite-forum120.htm>