Nonvolatile Field-Programmable Gate Array Using a Standard-Cell-Based Design Flow

Daisuke SUZUKI†Ⅰ and Takahiro HANYU†Ⅱ, Members

SUMMARY  A nonvolatile field-programmable gate array (NV-FPGA), where the circuit-configuration information still remains without power supply, offers a powerful solution against the standby power issue. In this paper, an NV-FPGA is proposed where the programmable logic and interconnect function blocks are described in a hardware description language and are pushed through a standard-cell-based design flow with nonvolatile flip-flops. The use of the standard-cell-based design flow makes it possible to migrate any arbitrary process technology and to perform architecture-level simulation with physical information. As a typical example, the proposed NV-FPGA is designed under 55nm CMOS/100nm magnetic tunnel junction (MTJ) technologies, and the performance of the proposed NV-FPGA is evaluated in comparison with that of a CMOS-only volatile FPGA. 

key words: FPGA, nonvolatile logic, logic synthesis, hardware description language, standard-cell-based design

1. Introduction

In the smart Internet of Things (IoT) era, a vast number of edge sever will be deployed to the proximity of IoT devices and data are efficiently processed; a paradigm that is now referred to as edge computing [1], [2]. In such edge computing, VLSI processors from low-end to high-end are indispensable for meeting a wide variety of application requirements. A field-programmable gate array (FPGA) is an effective way to implement such VLSI processors [3]–[8]. Because of the customizable hardware architecture, all its on-chip available logic blocks can be (re)configured with dedicated pipeline and parallelism design for performance (i.e. latency and throughput) enhancement and optimization. Especially, embedding the FPGA on a system-on-chip (SoC) [9], [10] is one of the most attractive candidates for IoT applications [11]. Recently, there are several research activities on a standard-cell based FPGA as flexible and portable option for embedded FPGA fabrics [12]–[17]. Commercial FPGAs are generally designed in a full-custom style (manual layout of transistors and wires) so as to optimize their speed performance, power and area. Such full-custom FPGAs are costly to develop and lack portability between process nodes. Moreover, full-custom FPGAs are difficult to embed into SoCs, owing to their fixed layouts and use of the full metal stack. In contrast, standard-cell-based FPGAs are described in a hardware description language (HDL) and can be pushed through a standard ASIC tool flow, making them straightforward to incorporate into a SoC in any arbitrary process technology. Fabric customization (e.g. domain specific hardware design) can be done by the modification of the HDL code, yielding greater design freedom when compared to modifying the full-custom layout. Meanwhile, standby power consumption in idle state is the most important issue in the conventional CMOS-only volatile FPGA. A nonvolatile FPGA (NV-FPGA), wherein all data are stored in nonvolatile devices, is one promising solution for the standby power problem [18]–[21]. Because the circuit information remains without a power supply, the standby power consumption is completely eliminated by utilizing a power-gating technique in which the power supply of idle circuit blocks is temporarily cut off. Despite of the attractive feature of the NV-FPGA, the tool flow of the NV-FPGA is custom made and closed, which prevents to demonstrate the objective advantage compared to the CMOS-only FPGA.

In this paper, a standard-cell-based NV-FPGA for ultra-low power embedded FPGA applications is presented. The NV-FPGA is designed and implemented by using a standard ASIC tool flow in conjunction with a nonvolatile flip-flop (NV-FF) [22]–[29]. The use of the standard ASIC tool flow makes it easy to perform system-level simulation, functional verification, architecture customization, and objective performance comparison with the CMOS-only FPGA. Moreover, the use of HDL also makes it easy to consider how to manage the control sequence of the FPGA components including circuit configuration, power gating, and so on. As a typical design example, an NV-FPGA is designed under 55nm CMOS/100nm magnetic tunnel junction (MTJ) [33]–[39] technologies and evaluated its benefits compared to the CMOS-only FPGA.

2. Overview of NV-FPGA

2.1 FPGA Architecture

Figure 1 shows the overall architecture of the FPGA which is composed of a 2-dimensional array of tiles [31]. The tile consists of a configurable logic block (CLB) with an array of basic logic elements (BLEs), a connection block (CB) to interface the CLB to the routing tracks, a switch block (SB) for signal routing among the routing tracks, and a controller for the circuit configuration. The BLE consists of a $K$-input

Manuscript received September 30, 2020.
Manuscript revised February 19, 2021.
Manuscript publicized April 16, 2021.
† The author is with the University of Aizu, Aizu-Wakamatsushi, 965–8580 Japan.
†† The author is with Tohoku University, Sendai-shi, 980–8577 Japan.
a) E-mail: daisuke@u-aizu.ac.jp
DOI: 10.1587/transinf.2020LOP0010
lookup table (LUT) circuit for a combinational logic whose truth table is stored in $2^K$-bit storage elements, an FF for sequential logic, and a multiplexer (MUX). The behavior of each function block is programmed by the configuration of storage elements. In the conventional CMOS-only FPGA, an SRAM cell is used for the storage element.

Figure 2 shows the routing architecture of the FPGA. A directional, single-driver wire is used in this paper owing to its smaller area and delay compared to those of a bi-directional wire [32]. Logic input and output signals of the CLB are interfaced by the CB and the signals are propagated to any adjacent tiles via the SB. The routing switch which is composed of a 1-bit storage element and an NMOS-pass gate is used for the signal routing in the CB. By storing logic ‘1’ into the storage element, the NMOS-pass gate is turned on and input and output signals pass through to/from the CLB. The SB is composed of four MUXs, four 2-bit storage elements, four input buffers, and four output buffers. By the configuration of the four storage elements, horizontal/vertical interconnects are routed at any four directions. We define the number of wires in the routing track as $W$, the ratio of wires in each channel to which a CLB input pin can connect as $F_{cin}$, and that of an output pin as $F_{cout}$. For example, in Fig. 2, $W$ is 4, $F_{cin}$ is 1 and $F_{cout}$ is 1, respectively.

Figure 3 shows the block diagram of the CLB which is composed of $N$ CLB slices. Each CLB slice has $I$ external logic inputs and $N$ internal feedback inputs from the CLB slices. The CLB slice consists of a $K$-input BLE and $K M$-to-1 multiplexers with storage elements. The output of the FF (Q) is selected for the output of the BLE (Z) when SEL=1, and the output of the LUT circuit (D) is selected when SEL=0, respectively. The LUT circuit performs any $K$-input logic function by using input vectors (Y). The $K M$-to-1 multiplexers interfaces between the LUT circuit and $M$ bit inputs ($I$ external inputs and $N$ feedback inputs).

Figure 4 shows the block diagram of the LUT circuit and its truth table which performs a 3-input combinational logic function. The truth table for the 3-input combinational logic function is stored in the 8-bit storage element. In the logic operation, the 3-bit external logic input ($X$) is applied to the 8-to-1 multiplexer and the content of one corresponding storage element is selected for the output. In this way, any arbitrary 3-input combinational logic function can be performed by the configuration of the 8-bit storage elements.

2.2 MTJ-Based NV-FF

Figure 5(a) shows the device structure of the MTJ device which consists of two ferromagnetic layers separated by
Fig. 4 3-input LUT circuit and its truth table.

| $Y_0$ | $Y_1$ | $Y_2$ | Z  |
|------|------|------|----|
| 0    | 0    | 0    | $M_0$ |
| 0    | 0    | 1    | $M_1$ |
| 0    | 1    | 0    | $M_2$ |
| 0    | 1    | 1    | $M_3$ |
| 1    | 0    | 0    | $M_4$ |
| 1    | 0    | 1    | $M_5$ |
| 1    | 1    | 0    | $M_6$ |
| 1    | 1    | 1    | $M_7$ |

Fig. 5 MTJ: (a) MTJ device, (b) $R - I$ characteristic.

To change $M$ from 0 to 1, the current which exceeds $I_{W_1-0}$ must be applied from the potted layer to the free layer.

Recently, a spin-orbit-torque (SOT) device which has a 3-terminal structure has become another attractive nonvolatile device [36], [37]. By employing the 3-terminal structure, the read property and the write property can be independently optimized which is quite useful for the circuit design [38], [39].

Figure 6 (a) shows a simplified block diagram of the NV-FF. A nonvolatile slave latch is connected to the volatile master latch. Figure 6 (b) shows a schematic diagram of the MTJ-based nonvolatile slave latch where an active load and a write driver are added to the standard CMOS latch [21]. During the normal mode, both LB and WB are set high and the slave latch operates almost same as a CMOS latch. Before power off, WB signal becomes low, and the write current driven by the write driver flows through two MTJ devices in accordance with the values of $q$ and $q_b$. Then, the temporal state of the CMOS FF is stored in the two MTJ devices in complementary fashion. After power on, LB signal becomes low and the stored data is recalled to the CMOS latch by sensing the current difference between the two MTJ devices. The active load is used for boosting the current difference. In this way, power supply can be turned on/off immediately without losing internal state. The physical layout of the NV-FF is designed by modifying the slave latch of the CMOS FF and adding hardware components (several logic gates and transistors) for store and recall operations [21]. Figure 6 (c) shows the cell layout of the MTJ-based NV-FF whose height is the same as other standard cells. The write driver and active load are divided into two parts and placed to the left and right sides of the CMOS-FF. In this paper, the MTJ-based NV-FF is added to the standard cell library and used for the storage element of the NV-FPGA.

Table 1 summarizes comparison of FPGA design methods. While the full-custom layout design requires high design cost and lacks design portability between process nodes, it is superior to standard-cell-based design in terms of speed, energy efficiency, and chip area. Therefore, the full-custom-layout-based FPGA is suitable for high-end-
standalone accelerators. While the standard-cell-based design is relatively higher delay, power, and area cost (e.g. 60% higher than that of full-custom design [15]), it is superior to the full-custom-layout-based FPGA in terms of design portability and flexibility owing to the HDL-based design. Therefore, the standard-cell-based FPGA is suitable for low-end-embedded accelerators. The standard-cell-based design is also superior in terms of ease of system-level verification and evaluation, and architecture customization by the use of standard EDA tools.

3. Design Flow of Standard-Cell-Based NV-FPGA

Figure 7 shows the tool flow for designing the proposed NV-FPGA. Synopsys Design Compiler is used for synthesizing the HDL code into the netlist of standard cells. Design compiler is also used for generating a standard delay format (SDF) file. By using the generated netlist and the SDF file, post-synthesis simulation including delay information of standard cells is performed. Cadence Innovus is used for performing place & route and generating physical layout of the synthesized netlist in GDSII format. The design rule check (DRC) and the layout-versus-schematic (LVS) are performed Mentor Calibre. Innovus also generates a post-layout netlist and an SDF file which includes delay information of parasitic components. For logic, post-synthesis, and post-layout simulations, Synopsys VCS is used. As described in the previous section, all the storage elements of the NV-FPGA is replaced by NV-FF. The NV-FF characterized by using supplementary design tools that make up for the lack of a precise simulation function of non-volatile device behavior. One of the tools is an MTJ-macro model which accurately reflects device parameters for the circuit simulation [45]. Another tool is Nanlib [46] which characterizes MTJ-based circuit IPs and its characterization data file is fully compatible with the common EDA file format. The physical layout of the NV-FPGA is drawn by using Cadence Virtuoso and the layout of CMOS FF. Virtuoso is also used for generating a layer-exchange format (LEF) file which defines physical dimensions of each standard cells and used in the Innovus flow.

To perform power gating, a power switch (PS) which is composed of an array of MOS transistors and interfaces between power line (VDD or GND) and circuit component

| Table 1 Comparison of FPGA design methods. |
|---------------------------------------------|
|                  | Full-custom-layout based | Standard-cell based |
| Ease of system-level verification and evaluation | -                  | +                  |
| Portability between process nodes            | -                  | +                  |
| Ease of customization                        | -                  | +                  |
| Design cost efficiency                       | -                  | +                  |
| Area efficiency                              | +                  | -                  |
| Speed                                        | +                  | -                  |
| Energy efficiency                            | +                  | -                  |
| Applications                                 | High-end-standalone accelerators | Low-end-embedded accelerators |

is needed. In the active mode, the PS is turned on while the PS is turned off in the standby mode. In the proposed NV-FPGA, each tile has a PS and the PS is individually turned on or off in accordance with the circuit configuration. Figure 8 shows the design flow of the PS. For the first step, the characteristic of the PS such as on current and off current are tuned by using HSPICE simulation. Then PS layout is drawn by using Virtuoso. After the DRC and LVS of the PS layout, parasitic components such as wire capacitance and wire resistance are extracted by Calibre and a netlist which includes parasitic component is generated. By using this netlist, more accurate characteristics of the PS can be evaluated by the HSPICE simulation.

Figure 9 shows the design flow of the proposed NV-FPGA. First, the RTL codes of FPGA components such as LUT circuits, CBs, and SB are described and its basic behavior is simulated by VCS. Then, the RTL code of tile is described by using the FPGA components. After the functional verification of the tile, logic synthesis and place & route is performed and the layout of the tile is generated. Then the PS is designed in accordance with the implementation result of the tile. Since the write current of the MTJ device is several hundred micro amperes, the peak current of the tile is proportional to the number of the NV-FF. Thus, the PS is designed so that sufficient write current is applied in the backup operation. Virtuoso is used for merging the layouts of the tile and PS and generating LEF file of the merged tile macro. By using the RTL code, the GDS and the LEF
files, the proposed NV-FPGA is designed, synthesized and its physical layout is generated.

4. Design Example

In this section, the NV-FPGA tile is designed using Verilog-HDL and its physical layout is implemented using a 55nm CMOS/100nm MTJ technologies. And as typical design examples, an 8-bit RCA and an arithmetic module with state machine are implemented on the array of $3 \times 3$ tiles. Their basic behaviors are confirmed by the VCS simulation with delay information obtained from the implementation result.

4.1 Design Environment

Table 2 summarizes design environment for the evaluation. Two types NV-FFs are used in the NV-FPGA. Thus, an NV-FF without set/reset is used for storing configuration data of each function block, and an NV-FF with synchronous reset is used in the BLE for implementing sequential logic. FFs in the other modules are not replaced by NV-FFs and used standard-CMOS FFs because their states do not need to be kept before power off.

4.2 Overall Architecture

Figure 10 shows the overall structure of the tile which contains 8 BLEs and 32 routing tracks with fully-connected CB ($F_{cin} = F_{cout} = 1$). Routing tracks are named as TI, RI, BI, LI, TO, RO, BO, and LO. They are connected to the routing tracks of the near neighbor four tiles. For example, TI is connected to the BO of the tile next to top. There is another inter-tile connections between the CB and the CLB, thus CLIL, CLOL, CLIB, CLOB, CBIR, CBOR, CBIT, and CBOT. For example, input port of the CLB namely CLIL is connected to the CBOR of the tile next to left. A controller which manages circuit configuration, nonvolatile FPGA specific operation, and a universal asynchronous receiver/transmitter (UART) to receive configuration data from external pad are embedded in the tile.

Figure 11 shows the state transition diagram of the controller whose initial state is Idle mode and the state is changed in accordance with the operation code (OPCODE). If OPCODE is CONFIG_CLB, the state is changed to CLB config. mode and circuit configuration to the CLB is performed. If the configuration is done (DONE=1), the state is returned to Idle mode. Similarly, configuration to the CB and the SB are performed when OPCODE is CONFIG_CB and CONFIG_SB, respectively. Because all storage elements are replaced by the NV-FFs and each data is stored in the CMOS part, it must be stored into the MTJ devices after the configuration. When the operation code is STORE_EN, the configuration data is stored into the MTJ device in each
NV-FF. To avoid large rush current, the store operation is performed at each function block in series. When OPCODE is OP_EN, the controller waits until the basic operation finish flag (FINISH) becomes high. When the basic operation is done (FINISH = 1), the NV-FF in each BLE stores own state to the MTJ device. Then power switch of the tile is turned off. After the power switch is turned on again and OPCODE is RECALL_DATA, the state of the NV-FF is restored from its own MTJ device. The use of RTL design is very useful to describe such complex control sequences.

Figure 12 (a) shows the hierarchy of the Verilog source code of the tile whose top module contains CONTROLLER, UART, and FPGA_TILECORE. FPGA_TILECORE is the main module of the tile and it contains four sub-modules, thus a CLB, two CBs (CB_H and CB_V), and SB. Figure 12(b) shows an example of the 3-input NV-LUT circuit (see Fig. 4). The module “NVSE” is used for the configuration memory cell of the CLB, the CB, and the SB. As shown in Fig. 12 (c), the standard cell of the NV-FF is directly instantiated in the Verilog code of the “NVSE.” Note that, FFs in CONTROLLER and UART are not replaced by NV-FFs (standard CMOS-FFs are used in these modules).

Meanwhile, a design issue for the standard-cell-based FPGA is how to implement the NMOS-pass gate in the CB. In this paper, the NMOS pass gate is replaced by the MUX as shown in Fig. 13(a). For the output of the CB (e.g., CBOT[0]), the MUX selects one input signal from \( W = 4 \) routing tracks. For the input of the CB (e.g., CBIT[0]), the MUX selects one output signal from CLB or passes through the input signal on the routing track. In this way, any arbitrary routing can be performed without the NMOS-pass gate. Figure 13(b) shows the Verilog source codes for implementing CB function which is mainly composed of several NV-SE and MUXs. The use of standard-cell-based design also makes it possible to individually optimize the drivability of the MUX and buffer at each routing track.

### 4.3 Configuration Examples

#### 4.3.1 8-bit RCA

As a typical configuration of the FPGA, an 8-bit ripple carry adder (RCA) is implemented on the array of the tiles. Figure 14 shows the 8-bit RCA implemented using two tiles. A full adder is implemented using two CLB slices, thus 4-bit RCA can be implemented in one tile and 8-bit one can also be implemented by cascading two tiles and propagating the fourth carry bit (CARRY[3]).

Figure 15 shows the configuration example of the 8-bit RCA on the FPGA with 3 × 3 tile array. TILE02 and TILE01 are used for implementing 8-bit RCA and the 8-bit
output (SUM[7:0]) propagates TILE12, TILE22, TILE21, TILE20, TILE10, and TILE00. The 3-bit XSEL and 3-bit YSEL signals selects tiles for configuration. In this case, TILE00, TILE01, TILE02 are selected and serial data DIN[0], DIN[1], and DIN[2] are serially written to the corresponding function blocks.

Figure 16 shows VCS simulation result of the 8-bit RCA using the synthesized netlist with delay information. We can confirm that corresponding outputs are obtained via routing tracks, CBs, SBs, and CLBs. Thus, functionality of the NV-FPGA is confirmed by using the synthesized tiles with delay information.

### 4.3.2 Arithmetic Module with State Machine

Another design example is an arithmetic module with a state machine which calculates \( S = A \times B + C \) as show in Fig. 17(a). When \( \text{set} \) becomes high, \( \text{state} \) becomes SET mode and the output signal \( \text{flag} \) becomes low. Then if \( \text{calc} \) becomes high, \( \text{state} \) becomes CALC1 mode. After \( \text{calc} \) becomes high again, \( \text{state} \) becomes CALC2 mode and it becomes IDLE mode in the next cycle.

Figure 17 shows the VCS simulation result of the arithmetic module with delay information. We can confirm that \( S = A \times B + C \) is collectedly calculated and \( \text{flag} \) becomes low just after \( \text{set} \) becomes high while \( \text{flag} \) becomes high just after \( \text{CALC} \) becomes high

### 4.4 Comparison with CMOS-Only FPGA

Figure 18 shows the layouts of the CMOS-only tile and that of the proposed tile. Due to the additional control circuit in the NV-FF, there is an area overhead in the proposed tile. Table 3 summarizes the comparison of three tiles. Thus, the CMOS-only tile, the CMOS-only tile with PS, and the proposed tile. While the CMOS-only tile without power switch is smaller than that of the proposed tile, it always consumes standby power consumption to keep internal data. By utilizing power gating, it can be completely eliminated. If power-gating technique is applied to the CMOS-only FPGA, all the data must be reconfigured to the volatile FF via the UART every time after power on which takes too long time. On the other hand, the wakeup time of the proposed tile is only 10ns because all the data are remained in the tile and no external

![Fig. 15](image1.png)  
**Fig. 15**  
Configuration example of 8-bit RCA using the array of 3 x 3 tiles.

![Fig. 16](image2.png)  
**Fig. 16**  
VCS simulation result of the 8-bit RCA using the synthesized tiles with delay information.

![Fig. 17](image3.png)  
**Fig. 17**  
Arithmetic module with state machine. (a) Block diagram its state transition diagram, and (b) waveform.

![Fig. 18](image4.png)  
**Fig. 18**  
Comparison of tile layouts.
data access is required. In terms of the dynamic power consumption, there is no significant difference between CMOS-only tile and the proposed one because the NV-FF operates the same as CMOS-based FF during normal operation.

Now let us discuss how to improve the area overhead of the proposed tile. Because the effective area of the MTJ-based NV-FF is more than two times larger than that of the volatile FF due to the additional hardware components for accessing the MTJ device. If the full-custom-layout-based NV-FPGA [20] in 90nm CMOS/MTJ hybrid technology is shrunk to 55nm CMOS/MTJ hybrid technology, the effective area is almost half compared to the proposed tile. A simple way to reduce the area overhead is to use more compact NV-FFs. For example, the use of the SOT device makes it possible to simplify the structure of the NV-FF and reduce the effective area [24], [29]. The use of the pass-gate logic also makes it possible to reduce the area because a CMOS MUX is replaced by several pass transistors. Another approach to enhance the area efficiency is to add some hardware macros to the standard cell library such as LUT circuits. Especially, we have already proposed a compact LUT circuit by using a logic-in-memory (LIM) structure where nonvolatile storage function and logic function are compactly merge into MTJ and MOS transistor network [40]–[44]. Moreover, the functionality of the LIM-based LUT circuit is almost compatible with that of conventional SRAM-based LUT circuit as shown in Ref. [43] and Ref. [44].

5. Conclusion

In this paper, a standard-cell-based NV-FPGA which is described in an HDL and can be pushed through a standard ASIC tool flow with NV-FFs has been presented. The use of ASIC tool flow makes it possible to migrate any arbitrary process technology and perform architecture-level simulation with physical information. As a typical design example, the NV-FPGA is designed under 55nm CMOS/100nm MTJ technologies and evaluated its performances compared to a CMOS-only FPGA. As a future work, we establish a tool flow for detailed power analysis for optimizing total energy consumption in IoT applications. Additionally, we extend the tool flow which can manage more complex nonvolatile logic circuit macros such as LIM-based LUT circuits and other multiple valued logic gates.

Acknowledgments

This research is supported by CIES consortium program, JST-OPEA Program Grant Number JPMJOP1611, JST CREST Grant Number JPMJCR19K3, and JSPS KAKENHI Grant No. 20K11725. This research is also supported by VDEC. We also would like to thank Y. Takako of the Focal Agency for his excellent technical assistance.

References

[1] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, “Edge computing: vision and challenges,” IEEE Internet of Things Journal, vol.3, no.5, pp.637–646, Oct. 2016.
[2] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skiee, “Mobile edge computing: a survey,” IEEE Internet of Things Journal, vol.5, no.1, pp.450–465, Feb. 2018.
[3] D. Bol, G. de Streel, and D. Flandre, “Can we connect trillions of IoT sensors in a sustainable way? a technology/circuit perspective (invited),” IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference, pp.1–2, Oct. 2015.
[4] S. Biookaghazadeh, M. Zhao, and F. Ren, “Are FPGAs suitable for edge computing?,” Proc. USENIX Workshop on Hot Topics in Edge Computing, pp.1–6, July 2018.
[5] A.R. Bucknall, S. Shrobejith, and S.A. Fahmy, “Network enabled partial reconfiguration for distributed FPGA edge acceleration,” Proc. IEEE Int. Conf. Field-Programmable Technology, pp.259–262, Dec. 2019.
[6] C. Hao, X. Zhang, Y. Li, S. Huang, J. Xiong, K. Rupnow, W.-M. Hwu, and D. Chen, “FPGA/DNN Co-Design: An efficient design methodology for IoT intelligence on the edge,” Proc. Design Automation Conf., pp.1–6, June 2019.
[7] G. Gobieski, B. Lucia, and N. Beckmann, “Intelligence beyond the edge: Inference on intermittent embedded systems,” Proc. Architectural Support for Programming Languages and Operating Systems, pp.199–213, April 2019.
[8] H.-Y. Ting, T. Giyahchi, A.A. Sani, and E. Bozorgzadeh, “Dynamic sharing in multi-accelerators of neural networks on an FPGA edge device,” IEEE Int. Conf. Application-specific Systems, Architectures and Processor, pp.197–204, July 2020.
[9] V.A. Ova and R. Saleh, “A “soft+” eFPGA physical design approach with case studies in 180nm and 90nm,” IEEE Computer Society Symp. VLSI, pp.1–6, March 2006.
[10] C.C. Wang, F.-L. Yuan, T.-H. Yu, and D. Markovic, “A multi-granularity FPGA with hierarchical interconnects for efficient and flexible mobile computing,” ISSCC Dig. Tech. Papers, pp.460–461, Feb. 2014.
[11] X. Zhang, A. Ramachandran, C. Zhuge, D. He, W. Zuo, Z. Cheng, K. Rupnow, and D. Chen, “Machine learning on FPGAs to face the IoT revolution,” Proc. IEEE/ACM Int. Conf. Computer-Aided Design, pp.894–901, Dec. 2017.
[12] J.H. Kim and J.H. Anderson, “Synthesizable FPGA fabrics targetable by the Verilog-to-Routing (VTR) CAD flow,” Proc. IEEE Field-Programmable Logic and Applications, pp.1–4, Sept. 2015.
[13] J.H. Kim and J.H. Anderson, “Synthesizable standard cell FPGA fabrics targetable by the Verilog-to-Routing CAD flow,” ACM Trans. Reconfigurable Technology and Systems, vol.10, no.2, pp.11:1–11:23, April 2017.
[14] B. Grady and J.H. Anderson, “Synthesizable heterogeneous FPGA fabrics,” Int. Conf. Field-Programmable Technology, pp.225–232, Dec. 2018.
[15] B. Grady and J.H. Anderson, “Physical design considerations for synthesizable standard-cell-based FPGAs,” Int. Symp. Highly-
[45] N. Sakimura, R. Nebashi, Y. Tsuji, H. Honjo, T. Sugibayashi, H. Koike, T. Ohsawa, S. Fukami, T. Hanyu, H. Ohno, and T. Endoh, “High-speed simulator including accurate MTJ models for spintronics circuit design,” Proc. IEEE Int. Symp. Circuits and Systems (ISCAS), pp.1971–1974, May 2012.

[46] M. Natsui, D. Suzuki, N. Sakimura, R. Nebashi, Y. Tsuji, A. Morioka, T. Sugibayashi, S. Miura, H. Honjo, K. Kinoshita, S. Ikeda, T. Endoh, H. Ohno, and T. Hanyu, “Nonvolatile logic-in-memory LSI using cycle-based power gating and its application to motion-vector prediction,” J. Solid-State Circuits, vol.50, no.2, pp.476–489, Feb. 2015.

Daisuke Suzuki received the B.E., M.E., and D.E. degrees from Tohoku University, Sendai, Japan, in 2004, 2006, and 2009, respectively. From 2010 to 2014, he was a Research Associate with the Center for Spintronics Integrated Systems, Tohoku University. From 2014 to 2015, he was an Assistant Professor with the Center for Innovative Integrated Electronic Systems, Tohoku University. From 2015 to 2020, He was an Assistant Professor with the Frontier Research for Interdisciplinary Research, Tohoku University. He is currently an Associate Professor with the University of Aizu. His main interests and activities are nonvolatile logic, its EDA tools, and edge devices using nonvolatile logic LSIs. Dr. Suzuki was a recipient of the Excellent Paper Award of The Institute of Electronics, Information and Communication Engineers of Japan in 2010.

Takahiro Hanyu received the B.E., M.E. and D.E. degrees in Electronic Engineering from Tohoku University, Sendai, Japan, in 1984, 1986 and 1989, respectively. He is currently a Professor and Education/Research Councilor (from April, 2018 to present) in the Research Institute of Electrical Communication, Tohoku University. His general research interests include nonvolatile logic circuits and their applications to ultra-low-power and/or highly dependable VLSI processors, and post-binary computing and its application to brain-inspired VLSI systems. He received the Sakai Memorial Award from the Information Processing Society of Japan in 2000, the Judge’s Special Award at the 9th LSI Design of the Year from the Semiconductor Industry News of Japan in 2002, the Special Feature Award at the University LSI Design Contest from ASP-DAC in 2007, the APEX Paper Award of Japan Society of Applied Physics in 2009, the Excellent Paper Award of IEICE, Japan, in 2010, Ichimura Academic Award in 2010, the Best Paper Award of IEEE ISVLSI 2010, the Paper Award of SSDM 2012, the Best Paper Finalist of IEEE ASYNC 2014, and the Commendation for Science and Technology by MEXT, Japan in 2015. Dr. Hanyu is a Senior Member of the IEEE.