Extended Abstracts of the 2020 International Conference on Solid State Devices and Materials, VIRTUAL conference, 2020, pp93-94

Systematic Design Flow for Realizing MTJ-Based Nonvolatile FPGAs

Yasuhiro Takako[1], Daisuke Suzuki[2], Masanori Natsui[1] and Takahiro Hanyu[1]

[1] Research Institute for Electrical Communications, Tohoku University
2-1-1, Katahira, Aoba-ku, Sendai, 980-8577, Japan.
Phone: +81-22-217-5508 E-mail: yasuhiro.takako.d1@tohoku.ac.jp
[2] School of Computer Science and Engineering, the Univ. of Aizu
7-3-1, Aizu Wakamatsu, Fukushima-ken, 965-8580, Japan
Phone: +81-242-37-2667

Abstract

Recently there are emerging activities for realizing high-speed and ultra-low-power logic LSIs such as field-programmable gate arrays (FPGAs) using nonvolatile storage elements such as magnetic tunnel junction (MTJ) device. However, a systematic design tool flow to support users' demands that automatically minimize the power dissipation under the timing constraints, has not been prepared until now. This paper shows a tool flow to systematically design nonvolatile FPGAs with automatic data-converting tools. It is also demonstrated that a simple logic circuit implemented using the proposed tool flow performs desired operation under logic simulation.

1. Introduction

A nonvolatile field programmable gate array (NV-FPGA) that uses magnetic tunnel junction (MTJ) as a storage element has the feature of reducing power consumption by executing power gating for each functional block while maintaining circuit information. In addition, the MTJ element can be stacked over the CMOS layer, so it can be expected to have lower power consumption and smaller size than ordinary static random-access memory (SRAM)-based FPGAs, and is expected to be widely applied to the application field as an edge application (Internet-of-Things; IoT) device [1]-[5]. However, since the current automatic design tools are limited to make only for ordinary SRAM-based FPGAs, the features of non-volatile FPGA (such as power gating control) are not taken into consideration. Therefore, in this paper, we launch a series of systematic design flow of automatic design tools suitable for non-volatile FPGA design. Specifically, given a desired logic circuit (Verilog HDL format) and FPGA architecture (configuration data), information about the FPGA hardware to be implemented (route, net, etc.) is automatically generated as the final data format (bit stream data).

2. Basic structure of nonvolatile FPGAs

Fig. 1 shows a basic structure of a nonvolatile FPGA that consists of a configurable logic block (CLB) that performs logical operations, a switch block (SB) that switches the vertical and horizontal wiring connections, a connection block (CB) that connects the CLB and wiring, and a power-management unit (PMU) that controls the power supply in each power domain.

SB connects the input/output of the surrounding CB, and CB determines the direction by connecting the input/output of the surrounding SB and CLB. The connection direction information is stored in MTJ elements.

The CLB contains several logic elements (LE), and the signals selected from the inputs from surrounding CBs and the outputs of multiple LEs including itself are used as inputs. The selection signal is stored in MTJ elements.

The LE is composed of a lookup table (LUT) for combinational circuit operation, flip-flop (FF) for latch, and multiplexer (MUX) for output selection. In the LUT, the logic input is applied to a MUX as a selection signal, and multiple programmed MTJ values are selected and output. The output goes to the MUX directly and to the MUX via FF. The FF output value is stored in the MTJ as needed. In MUX, two types of signals with or without FF latch are selected and output from LE.

3. Design example using the proposed tool flow

As a typical and simple example using the proposed tool flow, a 2-bit counter is designed. According to the flow in Fig. 3, a Verilog-formatted file that describes the circuit and an Architecture file that describes the configuration of the target FPGA are used for the input of Verilog to routing (VTR) [6].

VTR generates a file (.blif) which describes truth tables for the combinational logic circuits and sequential circuits used for implementing design target, and files (.net, .route, .place) showing the signal routing path as shown in Fig. 5. Since these files do not match the FPGA configuration format (bit stream), the conversion was done manually, which took a long time and many mistakes were made. However, the use of the proposed tool flow makes it possible to do it accurately and automatically in a short time as shown in Figs. 4 and 5. When Verilog simulation was performed using the bit stream output by the tool, correct operation is confirmed, so it can be said that the proposed automatic configuration-data creation tool has worked correctly as shown in Fig. 6.

4. Conclusion

In this paper, an automatic design flow to realize nonvolatile FPGAs has been shown and confirmed that as a circuit configuration example, when a user gave Verilog HDL data and FPGA configuration data, even the serial data required for FPGA implementation can be automatically generated. In the future, it is important to utilize the tool to search the optimal NV-FPGA architecture that achieves to minimize power consumption.

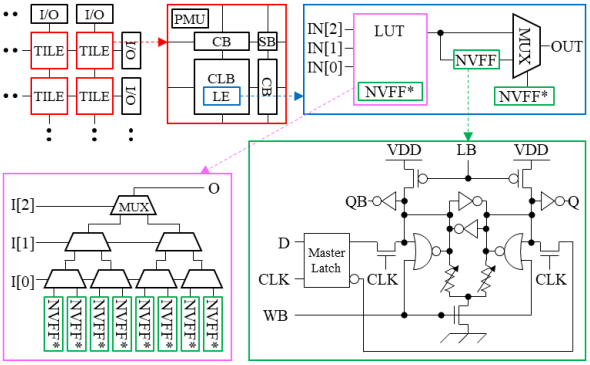
Acknowledgements

This research is supported by JST-CREST(JPMJCR19K3), JST-OPERA, JSPS KAKENHI Grant Number JP16H06300, CIES

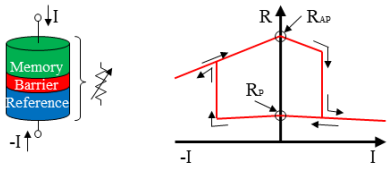
consortium program and VDEC.

References

[1] T. Hanyu, et al., Proc. IEEE, 104, 10, 1844/1863, Oct. 2016.
[2] D. Suzuki et al., Symp. VLSI Cir., C172/C173, June 2015.
[3] M. Natsui et al., IEEE JSSC, 54, 11, 2991/3004, Nov. 2019.
[4] N. Sakimura et al., IEEE ISSCC, 184/185, Feb. 2014.
[5] S. Fukami et al., ASP-DAC, 684/691, July 2014.
[6] J. Luu et al., ACM TRTS, 7, 2, 6:1/630, June 2014.



(a)



(b) (c)

Fig. 1 Configuration of non-volatile FPGA; (a) Overall configuration, LE, LUT, NVFF configuration, (b) MTJ structure, memory/barrier/reference, (c) MTJ waveform, high resistance (RAP) when a certain amount of current is applied in the + direction, low resistance (RP) when applied in the - direction. *NOTE) NVFF in LUT and MUX is modified using a Logic-in-memory circuit style[1].

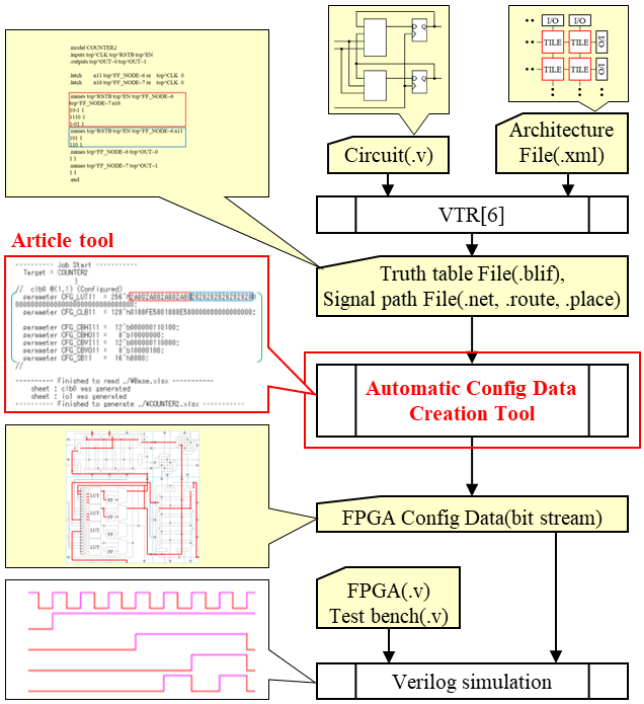


Fig. 3 Flow from circuit input to simulation verification.

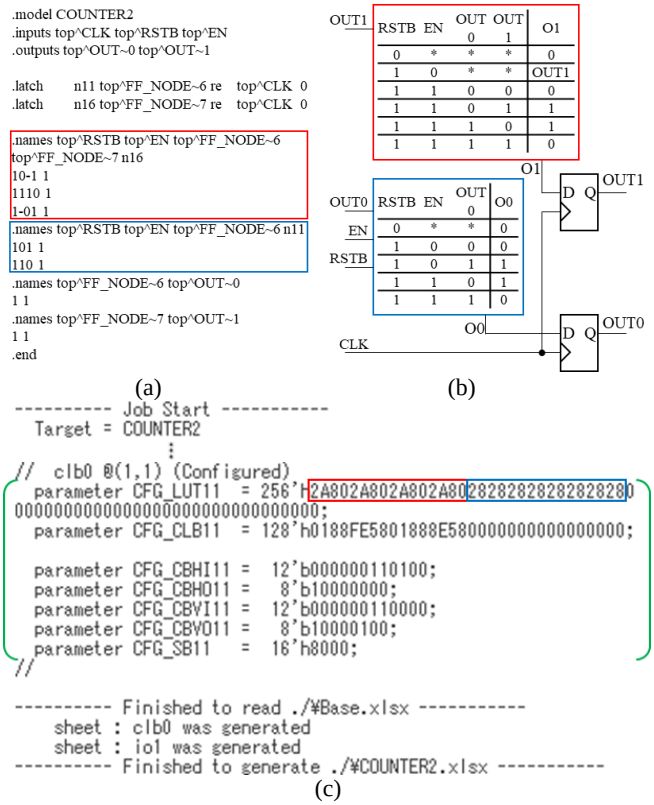


Fig. 4 Configuration tool in action, The red and blue boxes in (a) (b) (c) correspond to each other; (a) route file, Truth table and connection information. Truth table that is displayed only when the result is "1" , (b) Block diagram corresponding to (a), (c) Automatic Config Data Creation Tool execution log, Bit stream is in green brackets.

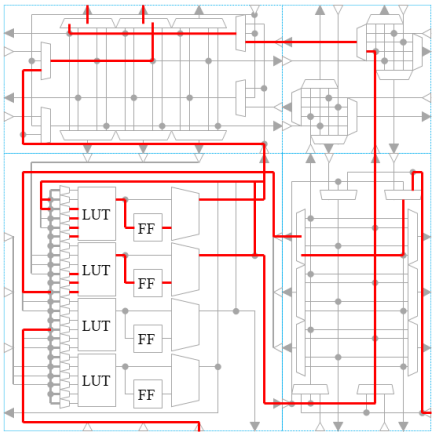


Fig. 5 Blocks surrounded by blue lines are CB (upper left), SB (upper right), CLB (lower left), CB (lower right), and red lines are bit stream signal path writing.

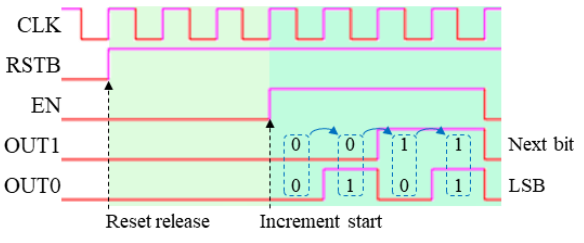


Fig. 6 Output waveform; OUT is incremented when RSTB="L", EN="H".